

(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD-A275 014



DTIC
ELECTE
JAN 27 1994
S **C**



THESIS

**DESIGN AND IMPLEMENTATION OF A DATABASE
MANAGEMENT SYSTEM FOR A NAVY MEDICAL
ADMINISTRATIVE UNIT**

by

Kevin Albert Bianchi
September 1993

Thesis Advisor:

Shu Liao

Approved for public release; distribution is unlimited.

94 1 26 046

94-02567



REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY	2. REPORT DATE 23 September 1993	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE DESIGN AND IMPLEMENTATION OF A DATABASE MANAGEMENT SYSTEM FOR A NAVY MEDICAL ADMINISTRATIVE UNIT		5. FUNDING NUMBERS	
6. AUTHOR(S) <i>BIANCHI, Kevin Albert</i>			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE *A	
13. ABSTRACT The Navy Medical Administrative Unit (NMAU) for the Monterey Peninsula has a challenging mission which encompasses many administrative tasks. Medical readiness and occupational health requirements are tracked for all Navy and Marine Corps personnel in the region. In order to fulfill their mission satisfactorily, it was necessary for NMAU to get an automated database management system. The Flight Surgeon at the Naval Postgraduate School works very closely with NMAU. The Flight Surgeon's administrative responsibilities would also benefit from a database system. Based on the requirements for NMAU a database system was designed and implemented in their clinic. Based on the Flight Surgeon's requirements the data base system was further analyzed in order to assist future upgrades that would employ the flight surgeon's requirements. The primary objective, however, was to get a system on line for NMAU that would enable them to effectively and efficiently execute their mission. The Navy Medical Administrative Unit Database System (NMAUDS 1.0) was the result of the previously described endeavors; it was written in dBASE IV version 1.5.			
14. SUBJECT TERMS NMAUDS version 1.0, dBASE IV version 1.5, Database Management System (DBMS), DBMS Design, DBMS Development, DBMS Implementation		15. NUMBER OF PAGES 169	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

Approved for public release; distribution is unlimited.

DESIGN AND IMPLEMENTATION OF A
DATABASE MANAGEMENT SYSTEM FOR A
NAVY MEDICAL ADMINISTRATIVE UNIT

by

Kevin A. Bianchi
Lieutenant, United States Navy
B.S., United States Naval Academy, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

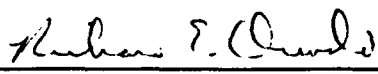
NAVAL POSTGRADUATE SCHOOL
September 1993

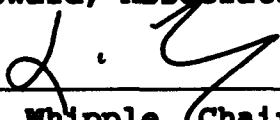
Author:


Kevin A. Bianchi

Approved by:


Shu Liao, Thesis Advisor


Richard E. Oswald, Associate Advisor


David R. Whipple, Chairman
Department of Administrative Sciences

ABSTRACT

The Navy Medical Administrative Unit (NMAU) for the Monterey Peninsula has a challenging mission which encompasses many administrative tasks. Medical readiness and occupational health requirements are tracked for all Navy and Marine Corps personnel in the region. In order to fulfill their mission satisfactorily, it was necessary for NMAU to get an automated database management system. The Flight Surgeon at the Naval Postgraduate School works very closely with NMAU. The Flight Surgeon's administrative responsibilities would also benefit from a database system. Based on the requirements for NMAU a database system was designed and implemented in their clinic. Based on the Flight Surgeon's requirements the data base system was further analyzed in order to assist future upgrades that would employ the flight surgeon's requirements. The primary objective, however, was to get a system on line for NMAU that would enable them to effectively and efficiently execute their mission. The Navy Medical Administrative Unit Database System (NMAUDS 1.0) was the result of the previously described endeavors; it was written in dBASE IV version 1.5

DTIC QUALITY INSPECTED 8

Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Date Recd	
Approved	
Date	
A-1	

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	BACKGROUND	2
B.	CHAPTER DESCRIPTIONS	4
II.	SYSTEM ANALYSIS	6
A.	USER SURVEY	6
1.	Methodology	6
2.	Application	7
B.	DATA FLOW DIAGRAMS	10
1.	Methodology	10
2.	Application	11
C.	REQUIREMENTS SPECIFICATION	12
1.	Data	12
2.	Reports	12
3.	Hardware and Software	14
III.	DATABASE DESIGN	17
A.	DATABASE CONCEPTS	17
B.	ENTITY-RELATIONSHIP DIAGRAM	18
1.	Methodology	18
2.	Application	20
C.	NORMALIZATION	22

D. DATA DICTIONARY	24
IV. DATABASE MANAGEMENT SYSTEM DEVELOPMENT	27
A. PROGRAMMING	27
B. TESTING	28
C. SYSTEM IMPLEMENTATION	31
D. SECURITY	32
E. MAINTENANCE	36
V. CONCLUSIONS	38
APPENDIX A: DATA FLOW DIAGRAMS	40
APPENDIX B: ENTITY RELATIONSHIP DIAGRAMS	46
APPENDIX C: DATA DICTIONARY	57
APPENDIX D: NMAUDS 1.0 PROGRAM CODE (PRG FILES)	69
APPENDIX E: USER'S MANUAL	147
LIST OF REFERENCES	160
BIBLIOGRAPHY	161
INITIAL DISTRIBUTION LIST	162

I. INTRODUCTION

This thesis designed and implemented a database system for the Navy Medical Administrative Unit (NMAU) at The Presidio of Monterey. NMAU supports all military units on the Monterey Peninsula with Navy and Marine Corps personnel assigned. The list of units include The Naval Postgraduate School (NPGS), Fleet Numeric Oceanographic Center (FNOC), Naval Security Group Detachment (NSGD), Naval Telecommunications Center (NTCC), Marine Corps Detachment (MCD), Personnel Support Detachment (PSD), Navy Research Laboratory (NRL), Defense Manpower Detachment Center (DMDC), and Branch Dental Clinic (BDC). The implementation of a database system greatly reduced the work hours spent on specific administrative tasks that are instrumental in accomplishing NMAU's principal task of medical readiness. The primary function of the database system is to maintain the necessary occupational health information on Navy and Marine Corps personnel. From this database standard reports are generated and ad hoc queries and reports are created.

The database design also considered the Flight Surgeon's functional requirements. Located at NPGS, the Flight Surgeon works closely with NMAU, as over seventy-five percent of the personnel they maintain records for are assigned to NPGS. The Flight Surgeon also performs the medical review portion of

physical exams on all Navy and Marine Corps personnel. This was done so that in the future the Flight Surgeon could have access to the database, thus assisting that office's reporting and query needs. The Navy Medical Administrative Unit Database System version 1.0 (NMAUDS 1.0) is written in dBASE IV version 1.5. NMAUDS is implemented on a MS-DOS-based personnel computer with future intentions to implement the database onto a Local Area Network (LAN) with the dBASE IV LAN version.

A. BACKGROUND

NMAU previously had a database designed with dBASE III in 1990 by the Officer In Charge (OIC) at the time. The dBASE III database system, in partial working order when this thesis began in January 1993, broke down and became incapable of performing any functions in April, 1993. Due to three facts the previous system had little use in the creation of NMAUDS 1.0. First, some requirements have changed since it's creation and there were no considerations in the dBASE III database system for future requirements flexibility. Second, there was very little documentation and therefore the code was very difficult to analyze. Third, it appeared to have many bugs as it was unable to accomplish its originally designed tasks.

NMAU presently maintains medical records for approximately two thousand Naval and Marine Corps personnel. The personnel encompass all possible Naval and Marine Corps career

descriptions and therefore have varying occupational health requirements. The challenging task of maintaining accurate readiness information is exacerbated by the following issues.

The commands that are supported are scattered about the Monterey Peninsula and are often in no way affiliated with each other. Personnel rotate through in very large numbers. An average of approximately three hundred people arrive and depart in the same period of time four times a year during academic quarters at NPGS and DLI. NMAU is a detachment from the Directorate for Community and Occupational Health (DCOH) at the Naval Hospital Oakland (NHA). Consequently NMAU support resources are over one hundred miles away. Finally, they operate out of the Medical Health Clinic on the Presidio of Monterey which is an Army base without a Naval headquarters.

Another unique characteristic of NMAU is that the bulk of the medical services were done, when this thesis began, by a civilian unit Primary Medical Care for the Uniformed Services (PRIMUS) based at the health clinic on the Presidio of Monterey. There is additional support from Silas B. Hayes Army Community Hospital on Fort Ord in Marina. Physical exams are conducted by PRIMUS with the exception of the medical review which is done by the Flight Surgeon. This arrangement changed beginning 01 August 1993 when the Army replaced PRIMUS with a detachment of doctors and medics from Silas B. Hayes.

There is one officer and six enlisted personnel (corpsman) allotted to NMAU. They are currently understaffed by one corpsman. The unit's purpose is to ensure that personnel get their medical requirements completed and that their records are properly documented. With eighteen hundred records to track, minimal support, a decentralized organizational structure, and a dynamic environment, the management challenge is noteworthy. This database system will enable NMAU to quickly list personnel deficiencies and status reports thereby saving countless work hours for the unit.

B. CHAPTER DESCRIPTIONS

Chapter II is a system analysis. The operating environment is studied by means of surveys followed by a specific diagraming process. A description of survey and diagraming methodologies are also provided. The chapter is concluded with requirements specifications as they pertain to data, reports, and hardware and software issues.

Chapter III is a summary of the design process followed while developing NMAUDS. The chapter begins with a small discussion of database concepts and how they are related to this system. Entity relationship diagrams are defined and the models created for this project are addressed. The normalization process is reviewed with respect to this database system. The final section in this chapter provides

commentary of the data dictionary and its benefits to database system design.

Chapter IV discusses the final phases involved with establishing a database system. These phases are the development portion of the process and they include programming, testing, and system implementation. Database security theory and how it applies to NMAUDS is discussed in the section following implementation and database maintenance issues concludes the chapter.

Chapter V is a conclusion. This conclusion does a short summarization of the thesis and addresses future modifications to the system created. Also included are lessons learned with respect to requirements for database management systems and how they change.

Appendices A-E supplement the previously described text. The appendices are; Data Flow Diagrams, Entity Relationship Diagrams, Data Dictionary, System Programs, and NMAUDS 1.0 Users Manual respectively.

II. SYSTEM ANALYSIS

A. USER SURVEY

1. Methodology

Surveys with the users are essential. No system can be designed without first understanding the current process intended for improvement. Users are defined in a database system as "people who need information from the database to carry out their primary business responsibility." (Hansen, 1992, p. 27) This includes everyone at the Navy Medical Administrative Unit (NMAU) in varying degrees. The primary questions one asks in the survey process are: "What are the functions and responsibilities of the organization? ...and... "What are the outputs and inputs?" (Whitten, 1989, p. 117) After this information is obtained determination of whether the process can benefit from a database system follows. If process improvement is possible the scheme for remodeling ensues.

The process includes formal and informal surveys. Formal surveys include scheduled appointments with predetermined questions. Informal surveys consist of touring the environment and asking questions about what is seen and what is not understood. Informal surveys continue throughout the system development process. In the system analysis for

this thesis, formal surveys were conducted with senior and experienced personnel and informal surveys conducted included all personnel.

2. Application

The primary concern of the personnel surveyed was the accuracy of the data. NMAU must be able to rely on the database to supply accurate reports to their supporting commands. Many individuals are unaware of their occupational health requirements in the Navy and Marine Corps. Those who are aware are often delinquent in maintaining these medical requirements. As a result, the notification of personnel for medical actions due determine how proper medical readiness is maintained. Ease of data input and error checking therefore become one of the most critical areas in designing the application.

Reporting requirements are basic. Reports about different medical deficiencies are supplied to the commands on a regular basis. There is, however, an additional necessity for flexibility in reporting. For instance, there exists a report that goes to various commands and curricular offices at NPGS which informs these units which personnel lack a current Physical Exam for the Personnel Fitness Test (PFT). The PFT, a semiannual requirement, contains no requirement for when execution of the PFT should occur. Hence, the report must be

flexible enough to assist individual commands or curricular offices with the PFT report in any given month as requested.

In addition to the many customized reports to various commands and curricular offices, reports must go to Directorate for Community and Occupational Health (DCOH) as well. These are known in Naval Medicine as "morbidity reports." A morbidity report and a productivity report have the same purpose. In other words, morbidity reports demonstrate what an occupational health unit accomplished over a period of time. For example, fulfilling monthly requirements to determine how many Yellow Fever Immunizations were given. Morbidity information is also comprised of ad hoc prospects.

Another administrative task that NMAU performs is tracking the number of people who arrive and depart each month. They are called "check-ins" and "check-outs" respectively. The previous database assisted this process and was invaluable to NMAU. Check-in and check-out statistics are necessary for Morbidity data as well.

During the interview process there was reference to the possibility of NMAU getting a Navy wide database system that might assist NMAU and perform some of their requirements. This system's title is Surface Automated Medical System (SAMS). The possibility of acquiring this asset was originally factored into future design considerations. However, later determination demonstrated SAMS use had fallen

short of NMAU's needs due to NMAU's unique situation previously discussed.

Security is a concern because of the fact that NMAU must share the health clinic spaces with PRIMUS and other medical units assigned to the health clinic. No data on the system is classified. There is, however, a degree of sensitivity with personal medical information. Equally as important is the desire to protect the data from malicious or unintentional corruption.

A final unusual factor taken into account involved the future of NMAU's responsibilities. As a result of down sizing and restructuring in the military, the fate of DLI, Silas B. Hayes and, in fact, any military unit is subject to change. There are many possible scenarios for these military units. The responsibilities of NMAU may increase or decrease. Presently, the most likely result is that NMAU's responsibilities will either increase or remain the same.

Most of the Flight Surgeon's requirements overlapped those at NMAU. In addition to those requirements, the Flight Surgeon had two other requirements in which a database would assist him. The first was more data requirements, including more immunizations, dental readiness information, rare billet designations called special programs, HIV testing and security clearance. The second revolved around the ad hoc nature of the job. The staff office at NPGS generated non-standard medical readiness information.

B. DATA FLOW DIAGRAMS

1. Methodology

Before designing a database it is necessary to accumulate the information from the surveys to study the data flows and gain an initial profile. From this research various diagrams are constructed in order to assist the design process. These are Data Flow Diagrams (DFDs).

Data Flow Diagrams are not flow charts. They do not explicitly show flow of data through a system. They only show flow of data, storage of data and the processes that respond to and change data. (Whitten, 1989, p. 14)

This is known as the logical schema of a system. The logical data flow eliminates unnecessary specifics such as who does a task and how a task gets done. The logical schema is only involved with data. This logical view of the system is derived from the physical data flow which was analyzed and determined as a result of the survey process (Appendix A).

There are three types of logical DFDs used (Appendix A includes a DFD symbols recognition key). The First is the Context Diagram. The Context Diagram portrays the system as one process and that one process' inputs and outputs. The Context Diagram also identifies the External Entities. External Entities are people, organizations or other systems which interact with the system being analyzed but fall outside the boundaries of the system identified in the Context Diagram. The second diagram is the Systems Diagram. This diagram identifies the subsystems and primary data stores.

They are depicted with their inputs and outputs as well. Finally there are Lower Level or n-Level Diagrams. Lower Level Diagrams represent the processes and data stores within a subsystem. This method of presenting more detail from one diagram to the next is called exploding. Each process can explore further into Lower Level Diagrams.

2. Application

The Context Diagram (Appendix A) for this system exhibits the primary functions of NMAU. The inputs are personnel medical records and requests for services. Three external entities, personnel or patients, supported units, and NHO receive the outputs.

The Systems Diagram (Appendix A) explodes the medical and occupational health services into three process and four data stores. The processes are Schedule Medical Service, NMAU Appointment and Flight Surgeon Appointment. The Data stores are Medical Records, Physical Exams Log, Immunization Log, and Flight Surgeon's Log. In order to determine the validity of a new system the new system should be compared to an accurate current system. Accordingly, the dBASE III has been taken out of the system because it became dysfunctional during the design process. Without the previous database NMAU's ability to complete their mission is greatly reduced. However, the Systems Diagram with dBASE III is depicted in Appendix A.

The only process exploded further is the NMAU appointments process (Appendix A). This is done to present greater detail in the report generation processes. New processes in this diagram are File Record, Provide Medical Services, Records Review and Create NMAU Reports.

C. REQUIREMENTS SPECIFICATION

1. Data

Identifying the data to be accumulated in the database was straight forward. Interviews were the primary determinant of essential data. However, often there are conditions in which the designer may anticipate how the system can benefit the user when the user can not. Therefore, some data fields, although not directly requested through interview, become a part of the design if acknowledged by the user as a future consideration. Notwithstanding, system designers must be careful not to over do their design.

The Data Dictionary (appendix C) is comprised of all the data fields required. Chapter III examines data dictionary methodology. The fundamental categories of data are patient personal information, medical history, and command information. The Flight surgeon's desires increased the number of fields in each category.

2. Reports

The reports can be broken into three classifications. They are Command Notification Reports, Internal Administrative Reports and DCOH Reports.

Command Notification Reports inform the supported units of individuals who are due or delinquent in fulfilling their medical and occupational health requirements. They include readiness reports and Personnel Lacking Current Physical Exams for the PFT or PFT reports. These reports have a few different variations depending on the period and parameters of the specific report.

Internal Administrative Reports are for house keeping within NMAU. They include check-in verification, check-out verification and a shelf list. Check-in verification includes verifying check-in data received from PSD. Check-out verification is necessary for preparation of personnel transfers but its primary use is a cross reference prior to removing data from the database or whatever mechanism is in use for tracking readiness. The shelf list displays the records in the sequence in which they are stored on the shelf. This is very useful for finding records or for the mandatory annual record review. The records are shelved by the last four digits of the social security number followed by the middle two digits succeeded by the first three digits. The Terminal Digit Filing System is the label given to this filing technique.

DCOH Reports deal primarily with morbidity. The necessary information is totaled on the database and transferred to an official report sent to NHO. The database may be authorized to generate this report in the future. This can be dealt with in a follow-on thesis by performing an upgrade of this application.

3. Hardware and Software

Often in systems design a portion of the design process includes selecting the hardware that best satisfies the specifications while staying within a budget. Sometimes the designer must design with hardware decisions previously determined. The latter was the case at NMAU. NMAU had an IBM compatible Zenith 286 as the only available asset for a database. They were requesting new and more powerful IBM compatible 386 or 486 computers with LAN capabilities. The decision to request these computers had already been determined and the approval of the request was pending when this thesis began.

As a result, hardware specifications were previously determined with a worst case scenario of designing a database to run off the old Zenith 286. The best situation to hope for was an approval of The IBM compatible 486. The decision made by DCOH in July, 1993 furnished NMAU with an Infiniti 486 IBM compatible computer.

The software requirements were quickly narrowed down to two different Fourth Generation Languages (4GLs). They were dBASE IV and Paradox. This decision resulted from the fact that at the time the Navy had site licenses for both of those packages. Therefore, they both were within budgetary constraints.

After review, Paradox emerged as the better relational language. It was designed for Structured Query Language (SQL) a powerful query tool. On the other hand, dBASE IV's strengths were its ease of use. The decision to use dBASE IV was a result of that fact. Furthermore, dBASE IV does have relational capabilities.

NMAU's mission was driven more heavily by periodic scheduled reports. Their need for ad hoc inquiries was minimal and the reduced relational potential of dBASE IV surpassed the requirement for ease of use. NMAU's personnel turn over every two to three years. The individuals assigned to NMAU have diverse computer experience but the norm is the more relatively inexperienced individual. Ease of use and wise ergonomics with respect to data entry are the paramount concerns since both software packages are capable of fulfilling all other requirements.

The Flight Surgeon's requirements are, in fact, primarily ad hoc driven. The ability to connect the Flight Surgeon into this database goes beyond the scope of this thesis and is an available topic for possible follow on thesis

work. Nevertheless, with respect to design the Flight Surgeon's requirements are analyzed. Two reasons guided this decision. First, if NMAU's responsibilities increase it is possible that their requirements would overlap further upon those of the Flight Surgeons. Additionally, dBASE IV version 1.5 has SQL capability and this application could be converted for the Flight Surgeon's use if desired.

III. DATABASE DESIGN

A. DATABASE CONCEPTS

A database is an integrated collection of data, stored with a minimum of redundancy and structured such that multiple applications can share the data. Ideally, the data is structured independent of the programs that use it. This allows the structure to be changed without having to change the existing programs that use the data. (Whitten, 1989, p. 65)

Or more simply, "A database ... is a collection of interrelated, shared, and controlled data." (Hansen, 1992, p. 33)

Databases can be centralized, one control processing unit (CPU) utilized in one geographic area, or distributed, many CPUs in a single geographic area or one or more CPUs employed in more than one geographic area. The term geographic in this context varies in definition. For the purposes of this thesis a geographic location will mean the same floor of the same building. In which case Navy Medical Administrative Unit database system (NMAUDS 1.0) will be implemented on a centralized system which will be modified, in fiscal year 1994, to a distributed system on a Local Area Network (LAN) and through thesis follow on may be distributed further geographically.

The components of a database system do not consist of hardware and software alone. Users, procedures, and data

stores are equally important components. These components combined become an input-process-output apparatus. Better said, the system takes in information, processes it and produces useful information or intelligence.

B. ENTITY-RELATIONSHIP DIAGRAM

1. Methodology

The entity relationship diagrams (ERDs) assist in the system analysis as well as design. This model's most useful attribute is that it exhibits the relationships between the objects that are in a system. Specifically those objects about which ERDs store data. ERDs are most useful when developing a relational database system in which there is routine query needs.

The ERDs provide a more detailed examination of the data stores in the Data Flow Diagrams (DFDs). A DFD presents data in its dynamic stages. The data stores in DFDs are the static pictures of data between their metamorphosis. The ERD presents a detailed analysis of the data stores thus depicting data at rest.

Entity relationship diagrams do not depict flow or processing. They should not be read like data flow diagrams or flow charts. Entity relationship diagrams depict data at rest, data being stored. They also do not imply how data is implemented, created, modified, used or deleted. (Whitten, 1989, pp. 229-230)

ERDs consist of three components entities, relationships and attributes. These components are

illustrated on the diagram as squares, diamonds and non-square rectangles respectively (Appendix B). The names given to Entities are nouns. Therefore, entities are persons, places or things. Things are more often referred to as objects but they also can be an occurrence known as an event. Some examples are patient, command, and medical record. An immunization is an example of an event.

Relationships have verb titles. Relationships associate entities with each other. These relationships fall into three categories: one-to-one, one-to-many and many-to-many. "Every patient has one medical record," is an example of a relationship between a patient and that patient's medical record. This is a one-to-one relationship in which "Has" is the name of the relationship.

Entities are the items which tend to have data stored about them. Descriptive elements that describe entities are attributes. Attributes are also known as data elements. Relationships can have attributes but on the ERDs of NMAU this does not occur. Fundamentally, the attribute-entity join is a relationship but the relationship is so elementary that it is more easily described as an attribute. "A medical record maintains yellow fever shot dates," is an example of an entity-attribute relationship. The relationship "maintains" and the entity "yellow fever shot date" possess no other pertinent data. As a result, it is better to combine the two

and designate "yellow fever shot date" as an attribute of medical record.

Each entity must have an attribute which uniquely identifies every entity instance. This attribute is commonly called the "key" attribute. The key attribute has an important function in the data dictionary tables discussed in section D. Some attributes may not have a value for each entity, these attribute instances are called null values.

2. Application

The ERD assembled for NMAU (Appendix B) deals primarily with personnel information. Medical Provider, Command, and Command-Division (departments, curriculums, or divisions) are three entities in the model whose data elements are presented predominately for future requirements. The remaining entities Patient, Medical History, Shot Record, Special Programs, and Dental Records have attributes which are necessary for current requirements. All eight entities are displayed with their attributes in Appendix B. The key attribute for all of these entities is Social Security Number (SSN) thus creating a data redundancy with the SSN element. These five entities can be combined into one entity but that picture yields less information about the data.

SSN is the key to the medical personnel entity as well. This would not be combined with the other entities whose key is SSN because there would be many null values since

only ten of the two thousand personnel in the database work at NMAU. In the future the database system can adopt this entity and its attributes to assist training and qualifications requirements of corpsmen.

Command Name is the key to the command entity. Unit Identification Code (UIC) and Curriculum Name are combined together to uniquely identify the Command Division entity. This is known as a combination or composite key. The Command and Command-Division entities, if implemented into the database system, will assist the flight surgeon's requirements. Within these two entities are the names of commanding officers and department/division officers. This induces a data redundancy which creates a tradeoff decision between the reward from increased information and the negative ramifications associated with this data redundancy. If implemented in an update of NMAUDS 1.0 this tradeoff will be a consideration.

A final point of interest for updates to NMAUDS is the possible addition of an entity called Female Tests. This entity comprises of the attributes Papanicolaou (PAP) smear and Mammogram. These tests are only performed on women. Since women incorporate far less than 50% of the records, if implemented, there would result an excess of null values. A relationship between Medical History and Female Tests (Appendix B) exists and can be considered in system updates if NMAU expands their readiness requirements. SSN would become

a redundant data item again and the tradeoff between reduced null values and increased data redundancy must be examined further prior to implementation.

C. NORMALIZATION

The normalization process ensures data is stored in a manner that minimizes data redundancy and data anomalies while maintaining data integrity. Data anomalies occur when deleted data precipitates an additional unintended data loss or when updated data does not consequently update related data. These anomalies are known as deletion anomalies and update anomalies respectively. Improved data integrity, or consistency of data, as well as reduced data redundancy facilitate the elimination of anomalies.

The normalization process consists of a series of rules which refine the data elements for database implementation. Each stage of the normalization process produces a data set in a form. As a result, First Normal Form (1NF) is the name of the first revision, Second Normal Form (2NF) the second, and the names of the remainder continue accordingly except for a couple of exceptions.

Normalization occurs concurrently with the creation of ERDs as the two procedures compliment each other. Analysis of entities and their attributes, as groups of data, is the method for viewing data. Every non-key attribute in an entity can be determined by the key. For instance, SSN determines

the value for last name, first name, sex and every other attribute in the patient entity. These correlations are called a functional dependencies or in other words the attribute last name is functionally dependent upon SSN. Functional dependency is an important concept to understand in normalization.

First Normal Form eliminates all data attributes which can have more than one occurrence for each entity. For example, in the entity Medical History the attribute Special Program might have two values for a single entity instance. That is, it is possible for personnel to be members of more than one special program. Therefore, in order to place Medical History into 1NF each individual program is formulated into an attribute.

Second Normal Form requires that the entity satisfy the criteria for 1NF and that no other data attribute in the entity be functionally dependent upon a portion of a composite key. Therefore, in the development of the Command-Division entity no attribute may be functionally dependent upon curriculum or UIC alone.

Third Normal Form (3NF) requires that the entity satisfy the criteria for 2NF and that no non-key attribute functionally determines another non-key attribute. Several data elements were eliminated as a result of this condition. Last physical exam in combination with flight and date of birth determine next physical exam. Consequently, next

physical exam was eliminated as an attribute. The previous example is one of a few cases which once corrected condenses the data entities into 3NF.

Boyce-Codd Normal Form (BCNF) is satisfied if every attribute that functionally determines any other attribute is a key. When satisfied this form accomplishes 2NF and 3NF as well. If data is already in 3NF the only condition remaining to examine is whether any portion of a combination key can be determined by a non-key. This will need further consideration if the Command-Division entity is adopted for use in an upgrade.

Fourth Normal Form (4NF) requires that entities satisfy the criteria for 3NF and that multivalued dependencies be eliminated. A multivalued dependency exists when there are more than two attributes in an entity, at least one attribute is multivalued, and the values of two attributes determine a third. The present version of NMAUDS does not have any multivalued dependencies.

Among others the remaining forms include Fifth Normal Form and Domain/Key Normal Form. These remaining forms consist of abstract concepts and would not divulge any additional weaknesses that would effect NMAUDS 1.0.

D. DATA DICTIONARY

Expansion, amalgamation with additional applications, and evolution are fundamental concerns in database system design.

In order to enable successful future modifications, the Data Dictionary was created. A data dictionary provides definitions of all the data items in the database. Definitions are organized in data dictionary tables. Data items include fields, records, and files. Files associated by file type are also assembled into tables along with a description.

In the past, organizations recognized that data was their most precious and uncontrolled resource. As databases and database systems proliferate data administration became a serious problem. (Whitten, 1989, p. 354) The creation of the data dictionary concept standardized data and data file definitions in an effort to correct data administration problems. The data dictionary now provides a reference to diminish data redundancy and assure data integrity.

A data dictionary is different from a project dictionary. Project dictionaries are directories of data elements, data flows, system requirements, system inputs and system outputs created during the design of a large database system. The project dictionary's function is to assist with the management of massive amounts of new definitions that can overwhelm designers if not organized. A project dictionary was not done for the NMAU system (NMAUDS 1.0) because of the relatively small size of the system. The effort and time invested in a project dictionary would not have yielded an equivalent payoff.

The data dictionary for NMAUDS is displayed in Appendix C. The Patient, Medical History, Shot Record, Special Programs, and Dental Record entities and relationships are organized into one data file for the active database system. The attributes which serve current requirements are maintained in this data file (Appendix C). Also included are some additional attributes anticipated for requirements in the near future. These attributes include Flu shot, Unit Identification Code (UIC), and Human Immunodeficiency Virus (HIV) test.

Possible future database file structures are depicted in the data dictionary as well. Currently, the requirement for ease of data entry has driven the decision to incorporate all of the data elements into one file. This is a result of the fact that dBASE IV data entry forms associate directly to a single database file. If the entities were grouped into two or more files than as many data entry forms would necessitate, thus reducing the efficiency of data entry.

IV. DATABASE MANAGEMENT SYSTEM DEVELOPMENT

A. PROGRAMMING

dBASE IV version 1.5 is a fourth generation language (4GL) and as a result, much of the programming is done by the application through standard formats. This proved very helpful with application, menu, indexing and query programs. However, some of the requirements for reports made it difficult to use the report forms in dBASE IV. Flexibility is often lost when using standard program forms. As a result, a good deal of time was spent learning the dBASE IV programming language, a third generation language (3GL), which is also available in dBASE IV.

All but two reports, "NPGS Personnel Without a Curriculum or Department Code on File at Medical" and "Records List in Shelf Order", are programmed in the dBASE IV 3GL. Programming concepts taught in ADA at the Naval Postgraduate School (NPGS) and in Fortran at the Naval Academy were the foundation of knowledge used. Additionally, Programming in dBASE IV, [Ref 3], aided the programming process. System programs can be found in Appendix D.

Most of the procedures creating reports query the operator for report parameters. While designing operator inquiries, safeguards were implemented to prevent errors. Almost every

query is in a verification loop, a multiple choice entry format, or a unsatisfactory data entry loop. Many are protected by more than one of the previously stated precautions.

B. TESTING

Testing is "the process of exercising or evaluating a system ... to verify that it satisfies specified requirements or to identify differences between expected and actual results." (Mynatt, 1990, p. 275) The testing phase of development overlaps substantially with the programming phase.

There are two principles of testing that were important to the testing strategy of NMAUDS. First, the purpose of the test plan is to uncover errors, not to demonstrate that the program can work. Second, one must recognize that it is impossible to test for all errors as there are an infinite number of possible data entries. Ergo, a test plan attempts to investigate as much of the program as possible with as little data as is reasonable.

Testing NMAUDS was done in the small and in the large. Testing in the small implies testing individual modules without respect to how they interact with other modules. The smaller modules were short programs, procedures, and functions. Testing in the large concentrates on passing test results between modules. Testing in the large included testing on report programs and applications.

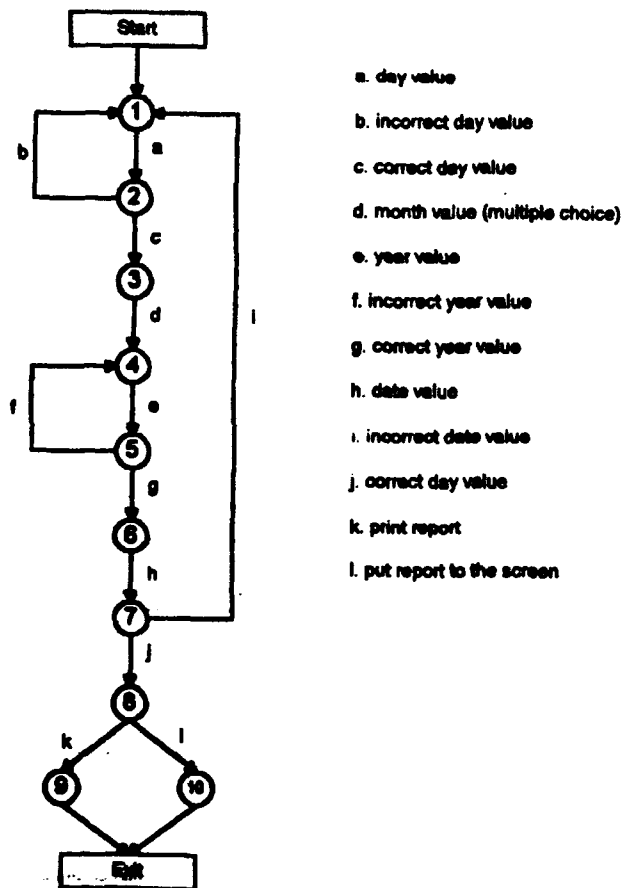
Black-box testing was done primarily in the small but was also performed in the large, while white-box testing was done solely in the large. White-box testing ensures every path within program logic is traveled. A Logic-Flow Diagram is constructed so that all passages through the program logic is illustrated. Figure 1 on the following page is an example of a Logic-Flow Diagram used in the test plan for the program Readins2 (Appendix D). Two passes through the program will cover all modules at least once (a-b-c-d-e-f-g-h-j-k and a-c-d-e-g-h-i-a-c-d-e-g-h-j-l).

Black-box tests are created using the technique of equivalence partitioning. To use equivalence partitioning, every input condition specified is divided into a number of equivalence classes. Each equivalence class consists of a class or set of data items all of which are similar to each other on some relevant dimension. To create test cases, test data are chosen that include at least one piece of data from each equivalence class. When these test cases are executed, it is assumed that, if the module performs correctly on the test item from a particular equivalence class, the module will perform correctly for any other item from the same equivalence class. (Mynatt, 1990, p. 292)

For example a black-box test on the module which establishes the day value of the readiness report, a-b-c in Figure 1, would create three equivalence classes. The range for the value is one to thirty-one inclusive. Therefore the equivalence classes are:

1. less than one
2. one through thirty-one
3. greater than thirty-one

White Box Test Plan Example



(Figure 1)

The final aspect of testing for NMAUDS occurred in implementation. The database system in its operating environment often gets tested in ways not thought of in the test plan. Additionally, the case in which test return values equal anticipated values but the anticipated values are erroneous, can only be discovered external to the test plan. Although not preferred, implementation will imminently discover some errors, preferably before acceptance. Only minor problems were discovered during the implementation of NMAUDS.

C. SYSTEM IMPLEMENTATION

NMAUDS was implemented in two stages. The first stage entailed of database file structure and data entry forms. This was accomplished in order to permit data entry to begin prior to the completion of NMAUDS. The NMAU mission was beginning to suffer greatly due to the total loss of their dBASE III system. NMAU estimated that they would need at least four weeks for data entry and verification. As a result, while the remainder of the system was still in development data entry for the database began. Implementation of the first portion of the system was done on the IBM compatible Zenith 286.

The second stage involved a more traditional system implementation. Final testing of the system was performed in these operations as was alluded to in the previous section.

The complete process included training NMAU personnel how to use the system, generating all reports for random cross checking of data produced with personnel medical records, and performing all maintenance and update functions of the database. The entire design was implemented six weeks after the initial implementation phase on an IBM compatible Infiniti 486.

There were minor corrections to be made in both phases of implementation. Neither stage manifested any serious problems hence, the implementation was deemed a success.

D. SECURITY

Computer and database system security is not limited to preventing access to unauthorized users. The following is a list of requirements for security of database systems created by Charles P. Pfleeger [Ref. 5:p. 304].

1. Physical database integrity
2. Logical database integrity
3. Element integrity
4. Auditability
5. User authentication
6. Access control
7. Availability

The contents of Pfleeger's list consists of those security concerns confronted in the design of NMAUDS. The database administrator is the person who monitors these security

issues. At NMAU the Officer in Charge (OIC) has the position of database administrator.

Natural disasters, power failures, intentional physical destruction, and unintentional physical damage such as dropping a disk pack are examples of physical database security concerns. Some of these events can not be prevented and as a result, easy back up procedures and back up reminders are installed in the application. The database administrator has been advised to make frequent backup copies of the database to a separate storage medium a standard operating procedure (SOP). The database administrator has also been encouraged to maintain backup copies of the system as well. Additional SOP suggestions to sustain physical security include: keeping food and beverages away from the computer, prohibition of smoking in the computer room, and keeping the computer behind a locked door when unattended.

Logical database integrity means the elimination of update and deletion anomalies. This is accomplished in part by dBASE IV design and buttressed further by the normalization process.

Protecting element integrity was performed in part by the desire to create an easy and isolated data entry environment. The dBASE IV data entry form further assists this concern through default values, multiple choice data entry, and restriction of undesirable characters within specified fields. Examples include default field entry of the current date for

the check-in field, multiple choice options for command, and restricting all characters other than numbers from the SSN.

Auditability, the ability to track file and record accesses, can be useful in recreating a sequence of events for determining security transgressions. This capability is not offered with dBASE IV. Presently, the cost of a system that would provide this capability would exceed the benefits gained by one. Thus, the accountability security measure consequently received no further attention.

The final three security interests are dealt with using the dBASE IV protect system. This protect system offers three levels of security log-in, file and field access, and data encryption. User authentication is satisfied with the protect system. The OIC has been briefed separately and Navy Medical Administrative Unit (NMAU) personnel have been instructed on proper password procedures. Included in, but not inclusive to the instruction were two particular points. First, users were told to avoid short length passwords and words that identify or associate users to their passwords when selecting a password. Second, they were also encouraged to avoid commonly used password storage methods which do not properly safeguard passwords.

Access control protection is offered through dBASE IV field and file access. Field access was not implemented because individuals with the lowest level of access privileges are the data entry personnel and they would need access to all

fields for data entry. File access levels fall into five levels as seen in the matrix in Figure 2 below.

<u>Access Privilege Levels</u>						
<u>Access Privileges</u>		Level #1	Level #2	Level #3	Level #4	Level #5
	Delete	Yes	No	No	No	No
	Extend	Yes	Yes	No	No	No
	Update	Yes	Yes	Yes	No	No
	Read	Yes	Yes	Yes	Yes	No

Security Access Matrix

(Figure 2)

The privileges associated with each level are on the vertical axis of the matrix. Availability will be directly controlled through the security levels depicted. The loss of necessary availability is another security issue in this final security list item. However, this will not be a factor until further applications are added to the system or new systems need to access the database.

Data encryption was deemed unnecessary due to the fact that all data items were unclassified. However, due to the private nature of most of the data, all of the data is considered sensitive. With all data elements designated sensitive, security is strengthened but flexibility is lost.

If another database management system design aspires to get access to the NMAUDS database, the higher level of security on NMAUDS would make the task more difficult. Finally, the OIC and NMAU personnel were instructed on the negative repercussions that can result from using unnecessary computer disks such as games and personnel text items on the same computer disk that the database and database system reside on.

E. MAINTENANCE

Maintenance is divided into four classifications: problem resolution, system modification, system expansion, and system upgrade or re-write. The last category is a combination of the first three. Problem resolution, if the system is tested and implemented properly, will account for the least amount of the total maintenance effort. System modification and system expansion will involve most of the maintenance labor, with estimates ranging between 70 and 85 percent.

NMAUDS' rigid design makes data input easy and system operation less intimidating for cyperphobic individuals. The tradeoff is that changes to, or maintenance of, the system is less flexible. Several reports, programs, and data elements of possible future requirements are installed in NMAUDS to avoid these maintenance efforts. Nonetheless, maintenance of some sort is inevitable as "every user's environment changes over time. The only difference is the rate of change. As change occurs, so do user requirements, and with change in

user requirements comes maintenance of the existing system."

- . (Inmon, 1983, p. 33) Effective program documentation and a thorough data dictionary are instituted in NMAUDS to assist these future projects.

V. CONCLUSIONS

Design, development and implementation of the Navy Medical Administrative Unit Database System (NMAUDS 1.0) is the result of this thesis. The NMAUDS system provides the Navy Medical Administrative Unit (NMAU) with information which assists their primary functions: maintaining occupational health data and ensuring that the commands they support sustain high levels of medical readiness. NMAUDS supports this mission by accomplishing medical record maintenance and medical and occupational health status reports in a efficient and effective manner.

During the period of time NMAU was without automated support their mission capability suffered. Once implemented, the system was immediately put to use. Future updates of the system are a possibility as medical requirements change or as the areas of responsibility change for NMAU.

Research of the Flight Surgeon's requirements was an additional focus of this thesis. A potential upgrade of this database system or a new database system sharing the NMAUDS database are plausible follow on projects as well. The data dictionary supplies valuable information to any future undertakings which expand or modify the system.

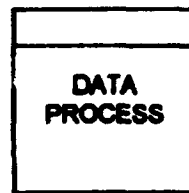
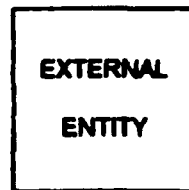
The tools provided in dBASE IV were useful and time saving. This fact should be true for any upgrades to NMAUDS

1.0. The manuals provided with dBASE IV were also quite useful especially in the development phase.

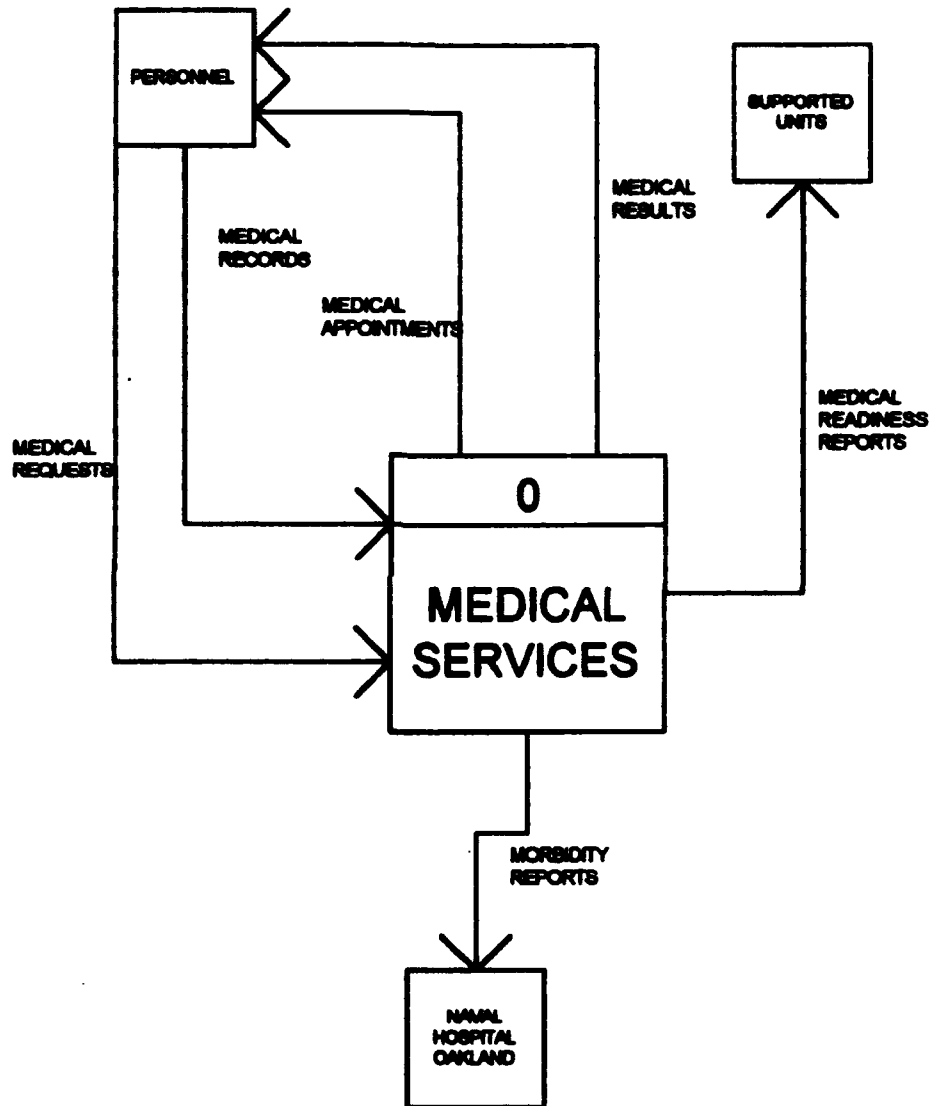
The commonly noted problem of changing requirements was a factor throughout the course of this thesis. Lessons learned while dealing with this certainty are worth noting. Requirements changing were due to four circumstances; the users' environment changes, the users recognize after demonstration of the technology new uses for the system, the users fails to disclose all of their own requirements to the designer, and the designer misinterprets the users' explanation of their requirements. Only the first item on this list can not be avoided. The latter three items on the list, although not completely avoidable, can be minimized with good communication with the users. In this thesis communication flow was sporadic and requirements changes resulted. In retrospect, periodically scheduled, possibly biweekly, progress meetings would have been advantageous.

APPENDIX A: DATA FLOW DIAGRAMS

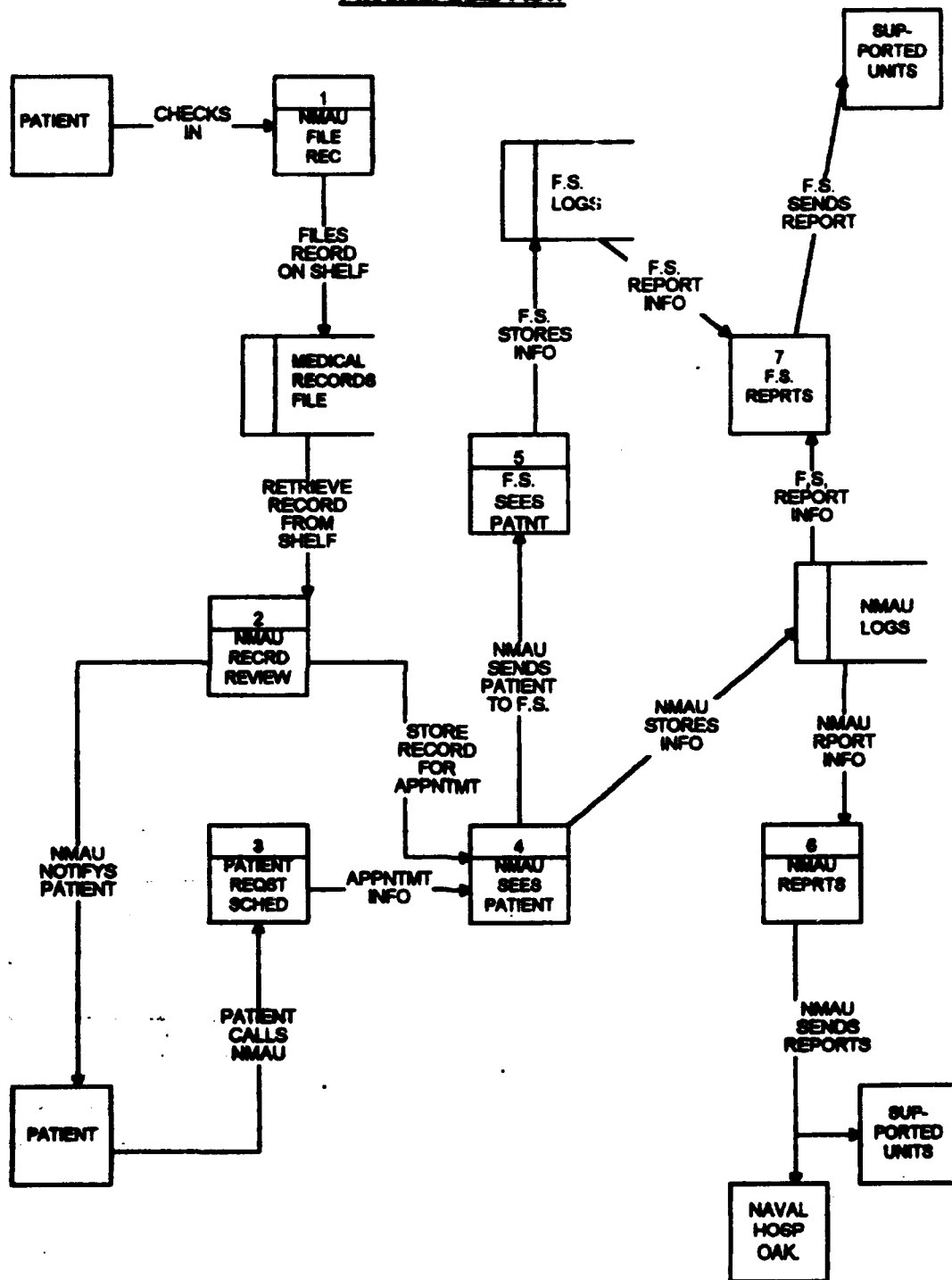
DFD SYMBOL KEY



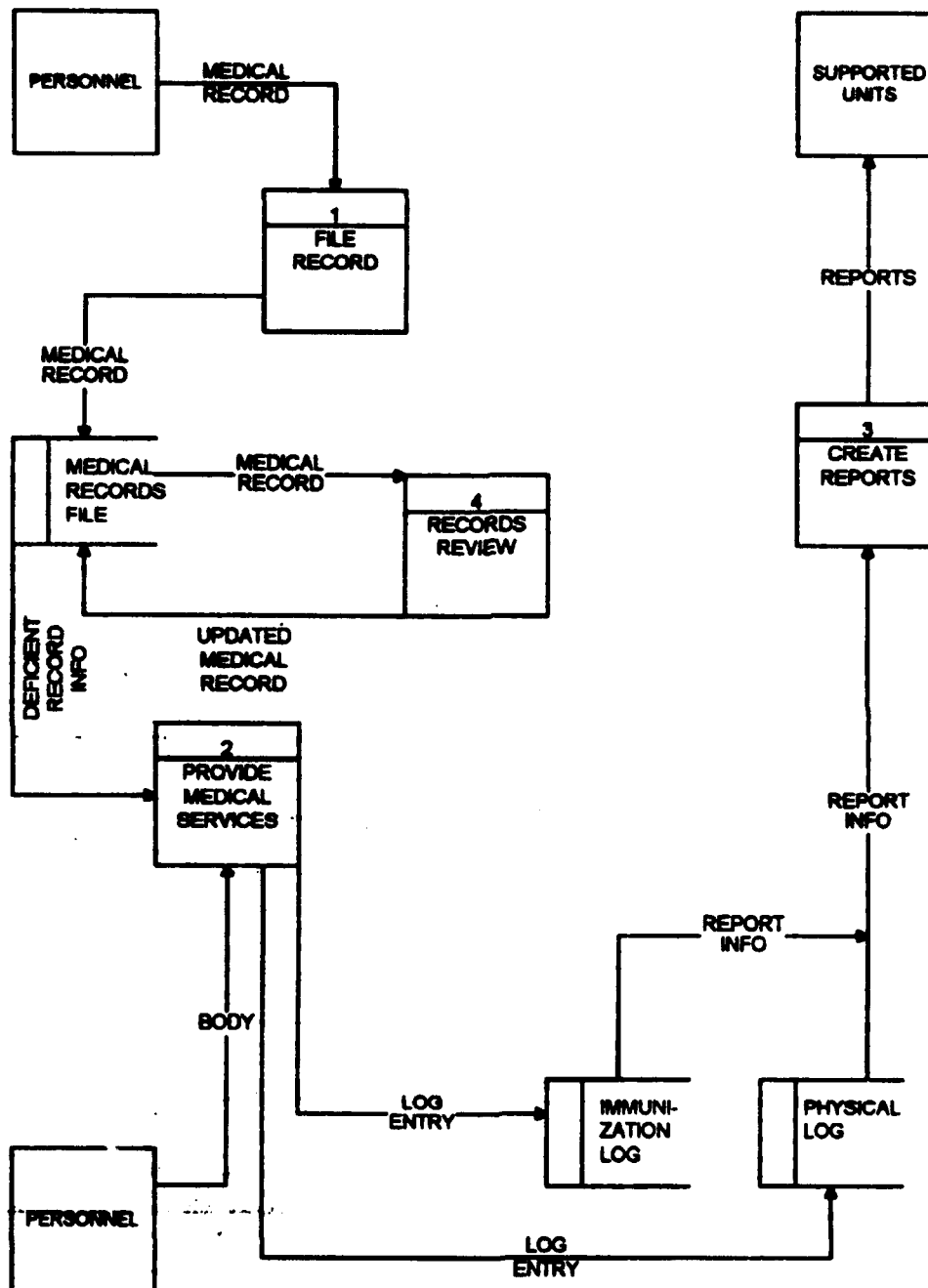
CONTEXT DIAGRAM



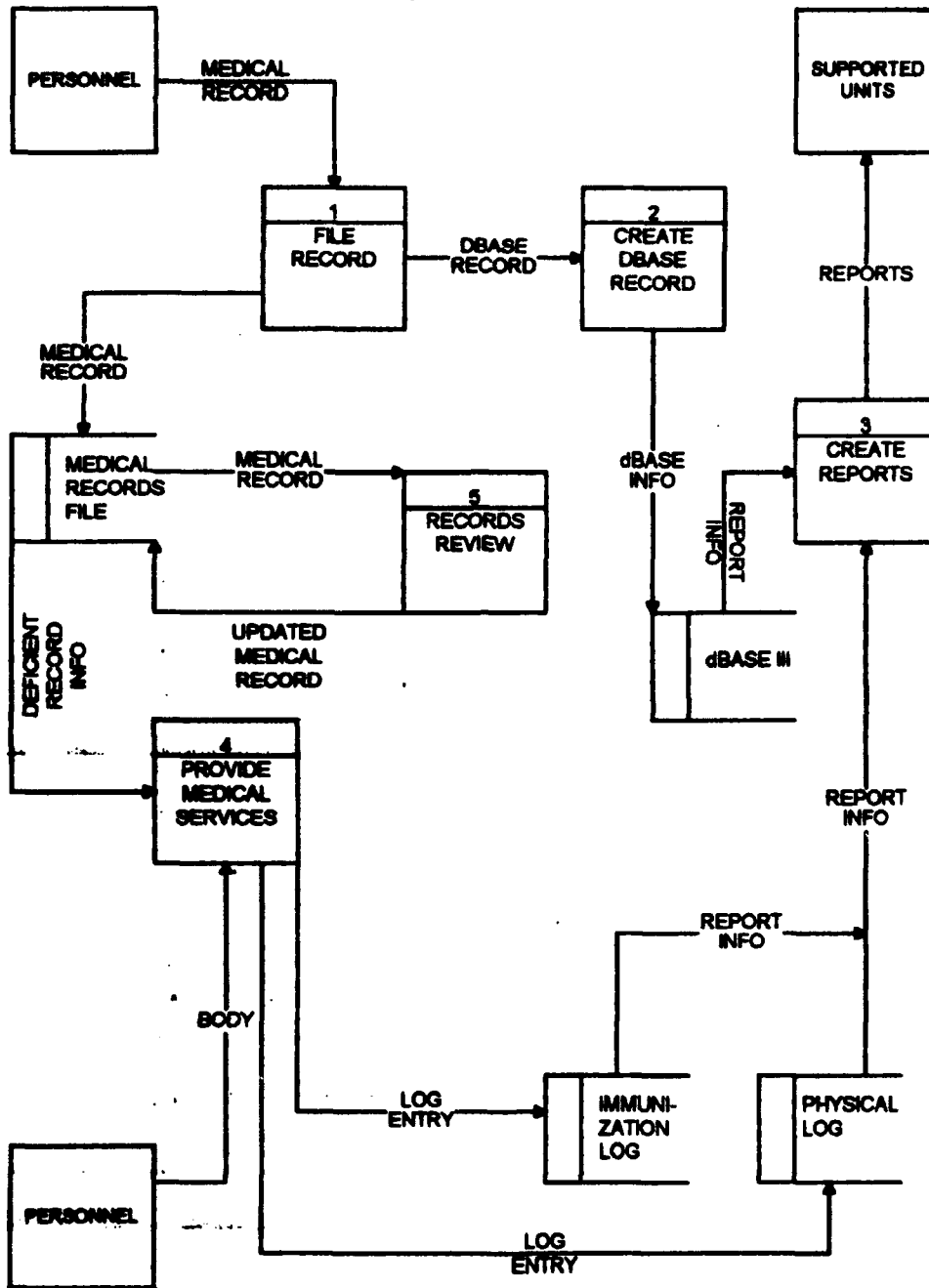
Physical Data Flow



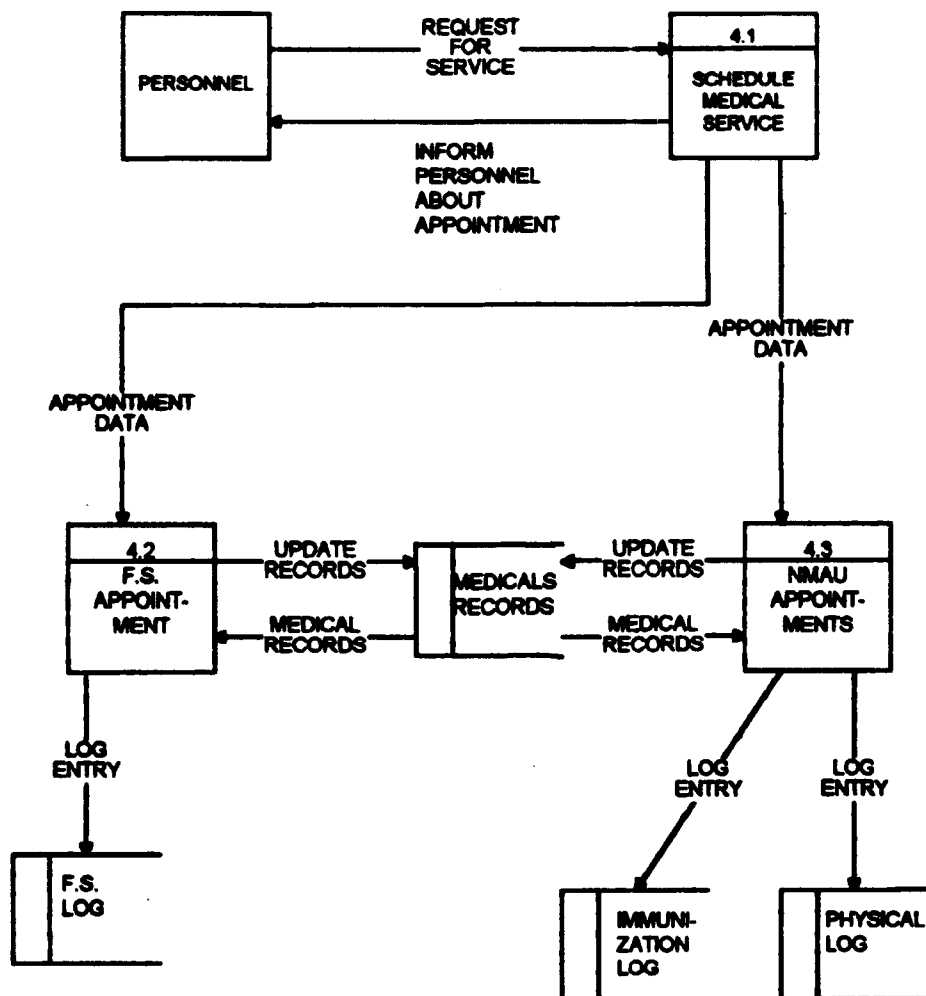
SYSTEMS DIAGRAM



SYSTEMS DIAGRAM (with dBASE III)

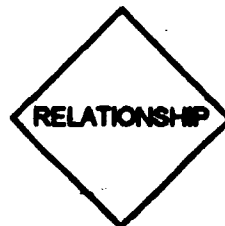
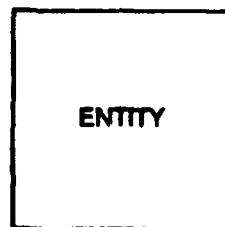


LOWER LEVEL DIAGRAM

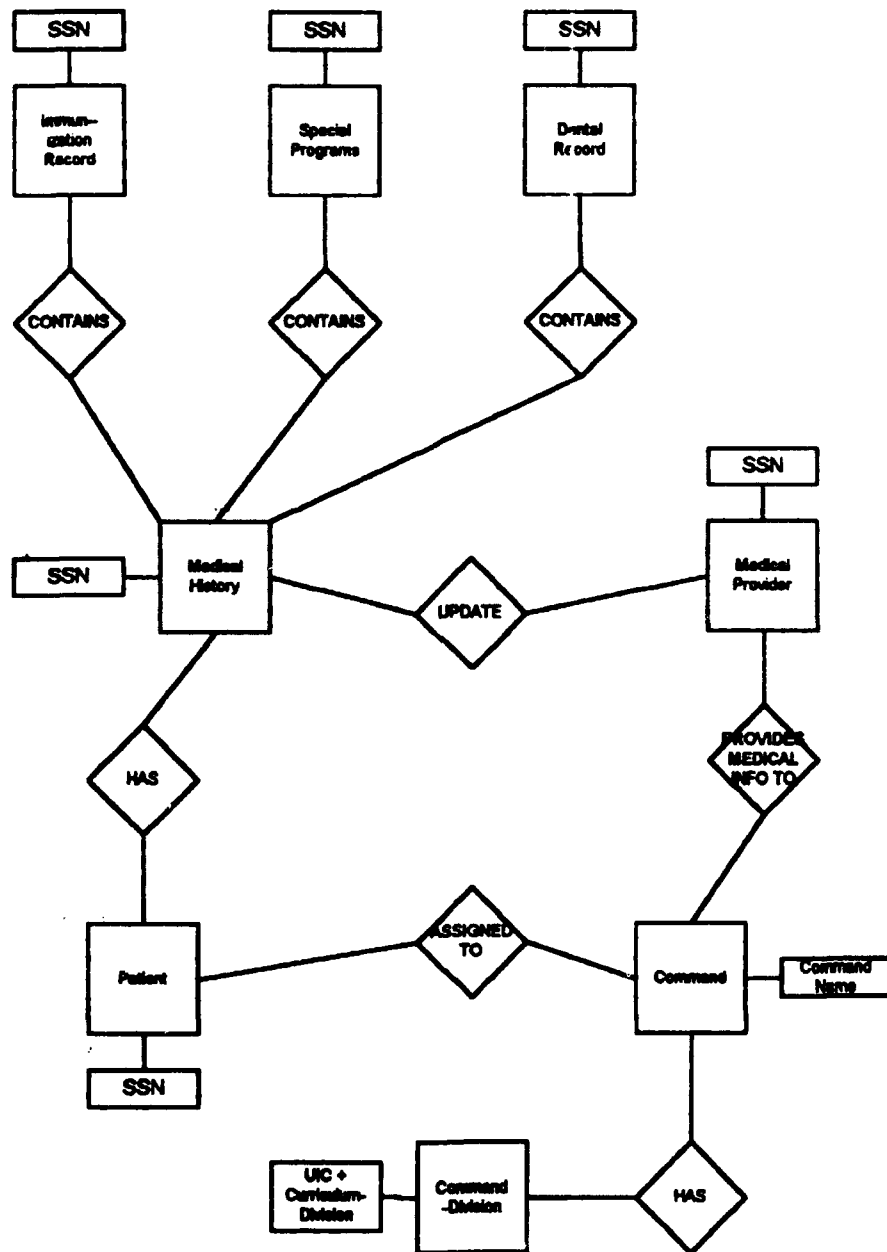


APPENDIX B: ENTITY RELATIONSHIP DIAGRAMS

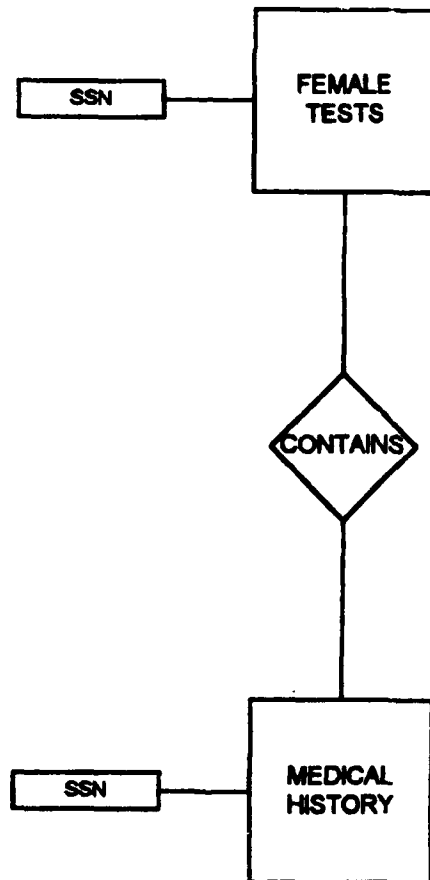
Entity Relationship Diagram Key



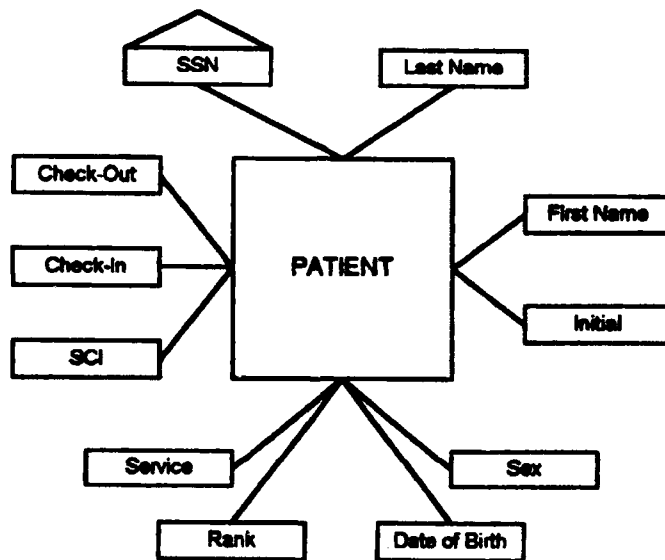
ENTITY RELATIONSHIP DIAGRAM



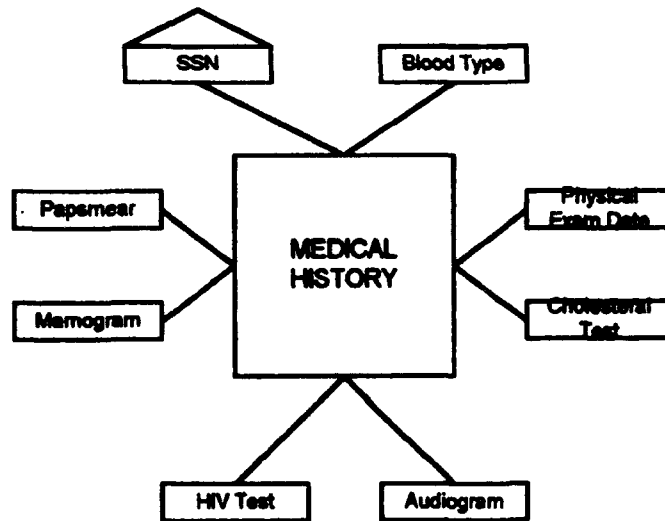
Female Tests - Medical History Relationship



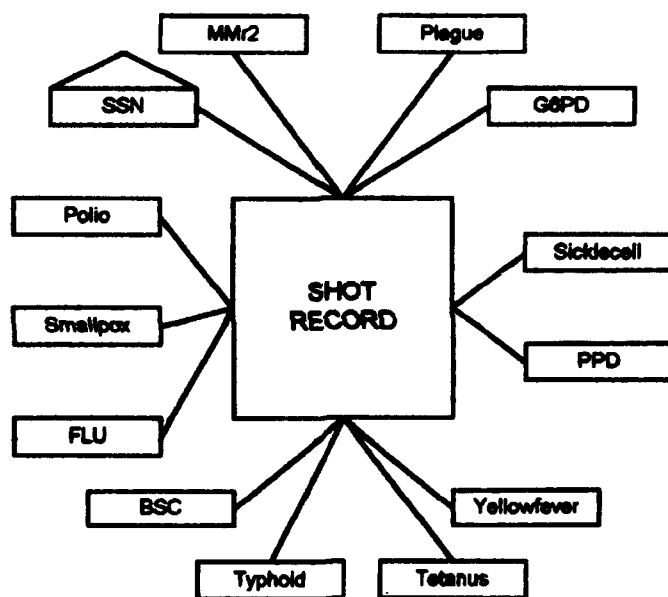
Patient Entity



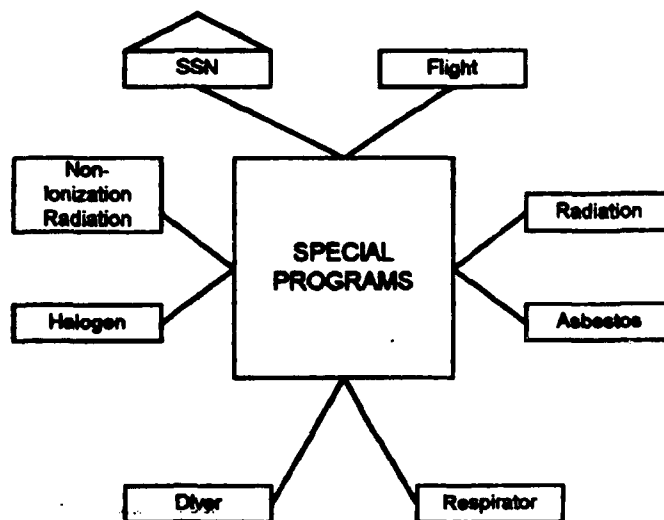
Medical History Entity



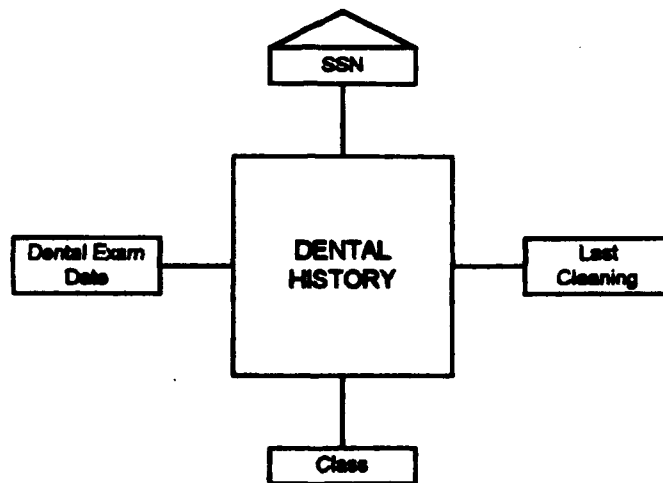
Shot Record Entity



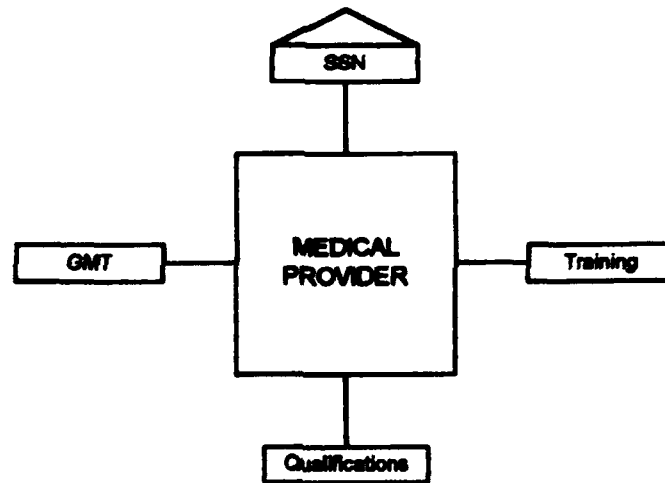
Special Programs Entity



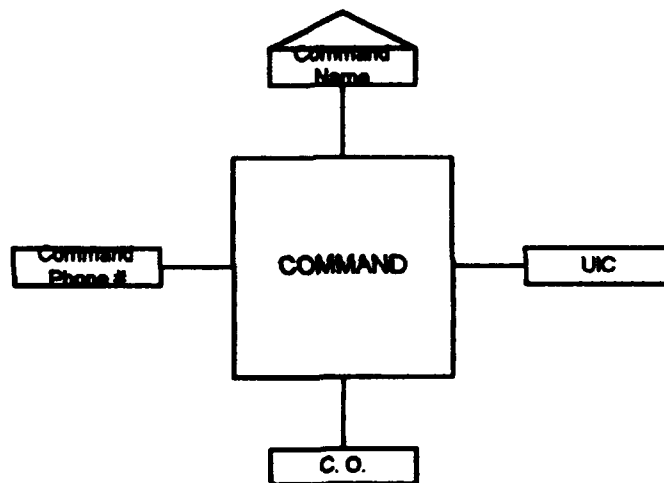
Dental History Entity



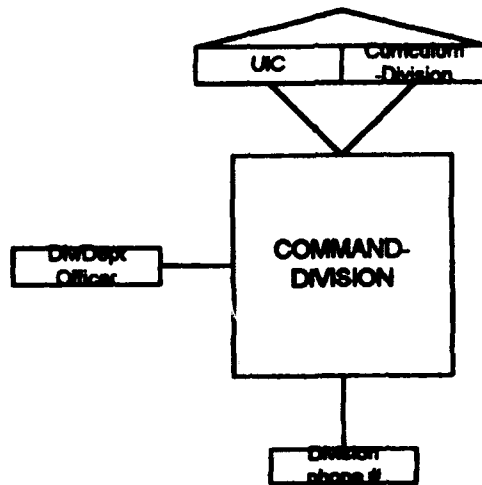
Medical Provider Entity



Command Entity



Command-Division Entity



APPENDIX C: DATA DICTIONARY

A. TABLE 1: DATA FILES

<u>FILE NAME</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
NEW_DATA.DBF	Data	Attributes about personnel and their medical record

POSSIBLE FUTURE DATA FILES

MED_HIST.DBF	DATA	Attributes pertaining to an individuals medical record
PATIENT.DBF	DATA	Pertinent attributes about an individual
CMND_DIV.DBF	DATA	Pertinent attributes about departments, divisions, or curricular offices (which to use will be driven by requirements)
COMMAND.DBF	DATA	Pertinent attributes about supported commands
DEN_HIST.DBF	DATA	Attributes about personnel dental records
MED_PROV.DBF	DATA	Attribute which assist with NMAU personnel administrative and training requirements

B. TABLE 2: NEW_DATA.DBF

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
SSNO	Character	11	Social Security Number
LASTNAME	Character	20	Last name of person
FIRSTNAME	Character	12	First name of person
INITIAL	Character	1	Middle initial of person
RANK	Character	3	Generic military rank, I.E. ensign or 1ST LT = "O-1"
SERVICE	Character	4	Military service USN or USMC
COMMAND	Character	4	Command name abbreviation
CHECKOUT	Date	8	Date a person will checkout of his/her current command
PHYSICAL	Date	8	Date last physical exam was completed.
BLOOD	Character	3	Blood type, "Due" is the value for unknown.
DOB	Date	8	Date of birth of the person
PPD	Date	8	Date the PPD immunization was given, 11/11/11 is entered for people who are allergic
YELLOWFEV	Date	8	Date the yellow fever immunization was given, 11/11/11 is entered for people who are allergic
TETANUS	Date	8	Date the tetanus immunization was given, 11/11/11 is entered for people who are allergic
TYPHOID	Date	8	Date the typhoid immunization was given, 11/11/11 is entered for people who are allergic
BSC	Logical	1	.T. if the typhoid basic series is complete otherwise .F.
AUDIO	Logical	1	.T. if an audiogram was performed during the last physical exam otherwise .F.
HIV	Date	8	Date of personnel's last HIV test.

B. TABLE 2: NEW_DATA.DBF continued

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
G6PD	Character	6	Glucose six phosphate dehydrogenase enzyme test; "Normal" if present, "Defcnt" if not present, "Due" if not performed
SICKLECELL	Character	3	Sickle cell test; "POS" if the trait is present, "NEG" if the normal, "Due" if not performed
FLU	Logical	1	.T. if a influenza shot was given otherwise .F.
POX	Logical	1	.T. if the smallpox immunization was given otherwise .F.
POL	Logical	1	.T. if the polio immunization was given otherwise .F.
FLIGHT	Logical	1	.T. if person is in a flight status otherwise .F.
RAD	Logical	1	.T. if person is in a radiation program otherwise .F.
ASB	Logical	1	.T. if person is in an asbestos program otherwise .F.
RES	Logical	1	.T. if person is in a respirator program otherwise .F.
DIV	Logical	1	.T. if person is in a underwater diver program otherwise .F.
HAL	Logical	1	.T. if person is in a high altitude low opening program otherwise .F.
NONION_RAD	Logical	1	.T. if person is in a non-ionizing radiation program otherwise .F.
CURRIC_DIV	Character	2	A code which uniquely identifies a curriculum or a division within a command
UIC	Character	5	Unit identification code, there can be more than one per command
CHECKIN	Date	8	Date person checked into NMAU

C. TABLE 3: MED_HIST.DBF

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
SSNO	Character	11	Social Security Number
PHYSICAL	Date	8	Date last physical exam was completed.
BLOOD	Character	3	Blood type, "Due" is the value for unknown.
PPD	Date	8	Date the PPD immunization was given, 11/11/11 is entered for people who are allergic
YELLOWFEV	Date	8	Date the yellow fever immunization was given, 11/11/11 is entered for people who are allergic
TETANUS	Date	8	Date the tetanus immunization was given, 11/11/11 is entered for people who are allergic
TYPHOID	Date	8	Date the typhoid immunization was given, 11/11/11 is entered for people who are allergic
BSC	Logical	1	.T. if the typhoid basic series is complete otherwise .F.
AUDIO	Logical	1	.T. if an audiogram was performed during the last physical exam otherwise .F.
HIV	Date	8	Date of personnel's last HIV test
CHOL	Date	8	Date of personnel's last cholesterol test

C. TABLE 3: MED_HIST continued

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
G6PD	Character	6	Glucose six phosphate dehydrogenase enzyme test; "Normal" if present, "Defcnt" if not present, "Due" if not performed
SICKLECELL	Character	3	Sickle cell test; "+" if the trait is present, "-" if the normal, "Due" if not performed
FLU	Logical	1	.T. if a influenza shot was given otherwise .F.
POX	Logical	1	.T. if the smallpox immunization was given otherwise .F.
POL	Logical	1	.T. if the polio immunization was given otherwise .F.
FLIGHT	Logical	1	.T. if person is in a flight status otherwise .F.
RAD	Logical	1	.T. if person is in a radiation program otherwise .F.
ASB	Logical	1	.T. if person is in an asbestos program otherwise .F.
RES	Logical	1	.T. if person is in a respirator program otherwise .F.
DIV	Logical	1	.T. if person is in a underwater diver program otherwise .F.
HAL	Logical	1	.T. if person is in a high altitude low opening program otherwise .F.
NONION_RAD	Logical	1	.T. if person is in a non-ionizing radiation program otherwise .F.

C. TABLE 3: MED_HIST continued

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
* PAPSMEAR	Date	8	Date of Personnel's last PAP smear test
* MAMOGRAM	Date	8	Date of Personnel's last mamogram test
PLAGUE	Date	8	Date the plague immunization was given, 11/11/11 is entered for people who are allergic
MMR2	Date	8	Date the measles, mumps, and rubella immunization adult series was completed

- * May be more advantageous as an attribute of the entity female tests.

G. TABLE 4: DEN_HIST.DBF

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
SSNO	Character	11	Social Security Number
CLASS	Numeric	1	Dental Classification
LAST_DE	Date	8	Date of last Dental exam
LAST_CLEAN	Date	8	Date of last cleaning

D. TABLE 5: PATIENT.DBF

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
SSNO	Character	11	Social Security Number
LASTNAME	Character	20	Last name of person
FIRSTNAME	Character	12	First name of person
INITIAL	Character	1	Middle initial of person
RANK	Character	3	Generic military rank, I.E. ensign or 1ST LT = "O-1"
SERVICE	Character	4	Military service USN or USMC
COMMAND	Character	4	Command name abbreviation
CHECKOUT	Date	8	Date a person will checkout of his/her current command
CHECKIN	Date	8	Date person checked into NMAU
PHYSICAL	Date	8	Date last physical exam was completed.
UIC	Numeric	5	Unit identification code, there can be more than one per command
CURRIC_DIC	Character	2	A code which uniquely identifies a curriculum or a division
SCI	Logical	1	.T. if the individual is cleared for sensitive compartmental information
DOB	Date	8	Date of birth of the person

E. TABLE 6: CMND_DIV.DBF

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
UIC	Numeric	5	Unit Identification Code
CURRIC_DIV	Character	2	A code which uniquely identifies a curriculum or a division
DIV_PHONE	Character	8	Division phone number
DIV_OFF	Character	11	SSN of the division officer

F. TABLE 7: COMMAND.DBF

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
COMMAND	Character	4	Command name abbreviation
UIC	Numeric	5	Unit Identification Code
CO	Character	11	Commanding Officer's SSN
CMND_PHONE	Character	8	Command phone number

H. TABLE 8: MED_PROV.DBF

<u>ELEMENT</u>	<u>TYPE</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
SSNO	Character	11	Social Security Number
TRAINING	Date	8	Date of last required training
GMT	Date	8	Date of last General Military Training
+ QUALS	Character		Qualifications

+ Probably will have to be expanded into several attributes once clarified in normalization.

I. TABLE 9: DATABASE INDEXES

<u>DATABASE FILE</u>	<u>PRODUCTION FILE</u>	<u>INDEX NAME</u>	<u>INDEX ORDER</u>
NEW_DATA.DBF	NEW_DATA.MDX	SSNO	SSNO
NEW_DATA.DBF	NEW_DATA.MDX	COMMAND	COMMAND, LASTNAME, FIRSTNAME, INITIAL
NEW_DATA.DBF	NEW_DATA.MDX	CURRIC	COMMAND, CURRIC, LASTNAME, FIRSTNAME, INITIAL

J. TABLE 10: REPORTS

<u>REPORT FILE</u>	<u>FILE VIEW</u>	<u>DESCRIPTION</u>
SHELF.FRG	SHELF.QBE	Medical records listed in terminal digit filing system order
NO_CURRIC.FRG	NO_CURRIC.QBE	List of personnel without a curric_div in the database
HIV_RPRT.PRG	N/A	Personnel without a current HIV test
MORBIDTY.PRG	N/A	Morbidity data for a requested month
OUT_RPRT.PRG	N/A	Personnel info for those marked for deletion
OUT_NEXT.PRG	N/A	Personnel info for those due to check-out in the month given
OUT_NOW.PRG	N/A	Personnel info for those in the database beyond their projected check-out data
PFT_REP.PRG	N/A	Personnel lacking a current physical exam as of the date of the report
PFT_REP2.PRG	N/A	Personnel who need a physical exam prior to a given date
PFT_REP3.PRG	N/A	Personnel who need a physical exam prior to a given date for the curric_div given
ONE_STOP.PRG	N/A	Readiness data for those marked for deletion from a given command
READINES.PRG	N/A	Readiness data due for a future time window
READINS2.PRG	N/A	Readiness data prior to a date given
READINS3.PRG	N/A	Readiness data prior to a date given for a given curric_div

K. TABLE 11: SUPPORT PROGRAMS

<u>FILE NAME</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
BACKUP.PRG	PROGRAM	Backs up new_data.dbf to newdatbk.dbf with production
CODE_NAM.PRG	PROCEDURE	Converts a curric_div to its associated text name
DUE_DATA.PRG	PROCEDURE	Print data for one_stop.prg
DUPECHK.PRG	FUNCTION	Disallows a repeated value for SSNO
HIV_CMND.PRG	PROCEDURE	Command heading for HIV reports
HIV_DATA.PRG	PROCEDURE	Print data for HIV reports
HIV_PGHD.PRG	PROCEDURE	Page header for HIV reports
HIV_SCHD.PRG	PROCEDURE	Screen page header for HIV reports
LOCATE.PRG	PROGRAM	Locates a record for edit/browse given a SSNO
OUT_CMND.PRG	PROCEDURE	Command heading for check-out reports
OUT_DATA.PRG	PROCEDURE	Print data for check-out reports
OUT_MARK.PRG	PROGRAM	Marks records for deletion given a SSNO
OUT_PGHD.PRG	PROCEDURE	Page header for check-out reports
OUT_SCHD.PRG	PROCEDURE	Screen page header for check-out reports
OUT_UMRK.PRG	PROGRAM	Un-marks records for deletion given a SSNO
PE_DUE_1.PRG	*FUNCTION	.T. returned if physical exam due by the current date
PE_DUE_2.PRG	*FUNCTION	.T. returned if physical exam due by the date value passed to the procedure
PE_DUE_3.PRG	*FUNCTION	.T. returned if physical exam due by the month after the current date
PFT_CMND.PRG	PROCEDURE	Command heading for PFT reports

K. TABLE 11: SUPPORT PROGRAMS continued

<u>FILE NAME</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
PFT_PGHD.PRG	PROCEDURE	Page header for PFT reports
PFT_SCHD.PRG	PROCEDURE	Screen page header for PFT reports
PHYS_DUE.PRG	*FUNCTION	Determines an individuals next physical exam due date
PPD_DUE.PRG	*FUNCTION	Determines an individuals next PPD immunization due date
PRN_RTRN.PRG	PROCEDURE	Returns system print variables after a print to screen
PRN_SCRN.PRG	PROCEDURE	Saves system print variables and prepares for a print to screen
RED_CMND.PRG	PROCEDURE	Command heading for readiness reports
RED_DATA.PRG	PROCEDURE	Print data for readiness reports
RED_PGHD.PRG	PROCEDURE	Page header for readiness reports
RED_SCHD.PRG	PROCEDURE	Screen page header for readiness reports
RESTORE.PRG	PROCEDURE	Returns new_data.dbf from newdatbk.dbf with production
TD_DUE.PRG	*FUNCTION	Determines an individuals next Tetanus immunization due date
TITLE.PRG	*FUNCTION	Determines an individual's service specific rank I.E. O-1 and Navy = "Ensign"
TYP_DUE.PRG	*FUNCTION	Determines an individuals next Typhoid immunization due date
YF_DUE.PRG	*FUNCTION	Determines an individuals next yellow fever immunization due date

*FUNCTION This type is a function (returns a single value) called as a procedure.

APPENDIX D: NMAUDS 1.0 PROGRAM CODE (PRG FILES)

***PRG files are listed in the order that they are displayed in the data dictionary.

```
*PROGRAM NAME:  HIV_RPRT
*PURPOSE       :  This is a report which displays:
*               1.  Name
*               2.  Rank(title)
*               3.  SSN
*               4.  Command
*               5.  Curriculum/division
*               6.  Date of last HIV test
*               of personnel who have not had their HIV test in
*               the last year.
*               --Individuals are grouped by their command.
*WRITTEN BY    :  Kevin Albert Bianchi
*LAST CHANGED:  8 AUG 93
```

```
SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER command
* New_Data DBF is used with command multiple index.
```

```
GO TOP
SET PRINTER ON
* Record pointer to the beginning of the DBF.
```

```
PUBLIC mcommand, ptitle, mlineno, title, mplength, mpageno,
selection
```

```
* Variables Initialized
MCOMMAND = ""
_pageno = 1
ptitle = "PERSONNEL LACKING CURRENT HIV TEST"
mlineno = 0
mplength = 60
mpageno = 1
selection = SPACE(1)
* Variables initialized
```

```
@ 5,10 SAY "Do you want the report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."
@ 10,35 GET selection PICTURE "@M P,S";
MESSAGE "Press SPACEBAR to scroll, Enter to select"
```

```
READ
CLEAR
```

* Assigns a value to selection which runs a print job or a put to the screen.

```
IF selection = "P"
  PRINT JOB
  DO HIV_PgHd
  * Calls the page header program.
  DO WHILE .NOT. EOF()
    MCOMMAND = COMMAND
    mcount = 0
    DO WHILE COMMAND = MCOMMAND
      IF mlength = mlineno
        mlineno = 0
        EJECT PAGE
        DO HIV_PgHd
        DO HIV_Cmnd
        * No Cmnd procedure written specifically for this
        report as the
        * command heading is the same for both reports
        upon initial design.
      ENDIF
      * Line # counter to zero, page advanced, Header
      programs called.

      IF CTOD(STR(DAY(HIV)) + "/" + STR(MONTH(HIV)) + "/"
      + STR(YEAR(HIV)+1)) <= DATE()
      * Determines if an HIV test has been done in the last
      year.

      mcount = mcount + 1
      IF mcount = 1
        DO HIV_Cmnd
      ENDIF
      * Prints command header when a new command is
      encountered.

      DO TITLE
      DO HIV_DATA
      * Output programs called
    ENDIF
    SKIP
  ENDDO
  * Continue while the command remains the same.
ENDDO
* Continues until EOF marker is located.
EJECT PAGE
ENDPRINTJOB
ELSE
  DO PRN_SCRN
  * Procedure to prepare print variables for a put to the
  screen.
```

```

PRINTJOB
DO HIV_Schd
* Calls the page header program.

DO WHILE .NOT. EOF()
  IF _plength = mlineno
    mlineno = 0
    EJECT PAGE
    DO HIV_Schd
  ENDIF
  * Line # counter to zero, page advanced, Header programs
  called.

  IF CTOD(STR(DAY(HIV)) + "/" + STR(MONTH(HIV)) + "/" +
    STR(YEAR(HIV)+1)) <= DATE()
    * Determines if an HIV test has been done in the last
    year.

    DO TITLE
    DO HIV_DATA
    * Output programs called
  ENDIF
  SKIP
ENDDO
* Continues until EOF marker is located.
ENDPRINTJOB
DO PRN_RTRN
* Procedure to return print variables to their original
values.
ENDIF
SET ESCAPE ON

```

```

*PROGRAM NAME:  MORBIDTY
*PURPOSE      :  This is a report which displays morbidity data
*               for a specified month.  Data items include
*               check-ins, checkouts, yellow fever, PPD,
*               tetanus, typhoid and physical exams.
*               The program will query the individual for the
*               month.
*               --Individuals are grouped by their command.
*WRITTEN BY   :  Kevin Albert Bianchi
*LAST CHANGED:  28 JUL 93

```

```

SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER command
* Uses new_data DBF and the command multiple index

GO TOP

```

*Record pointer to the top of the DBF printer set on.

```
PUBLIC mcommand, mlineno, report_month, report_year,
report_date,; ptitle, title, mplength, mpageno, print_job
* variables declared global
```

* Variables initialized

```
print_job = SPACE(1)
mlineno = 0
mpageno = 1
mplength = 60
report_month = SPACE(2)
report_year = 1993
report_date = {}
physical_count = 0
checkin_count = 0
checkout_count = 0
yf_count = 0
ppd_count = 0
td_count = 0
typ_count = 0
```

* Variables initialized

```
@ 5,4 SAY "This report will list the morbidity information for
the month selected."
```

```
@ 10,5 SAY "Enter the number equal to the month of your report
and press return."
```

```
@ 18,35 GET report_month PICTURE "@M
01,02,03,04,05,06,07,08,09,10,11,12";
MESSAGE "Press SPACEBAR to scroll, Enter to select."
```

READ

CLEAR

* Multiple choice query for report month.

```
@ 5,10 SAY "Enter the year which contains the month of this
report."
```

```
@ 7,11 SAY "Enter all four numbers (I.E. For 1993 ENTER
1993)."
```

```
@ 12,35 get report_year PICTURE "9999"
```

READ

CLEAR

* Query for year of report.

```
report_date = CTOD("01/" + report_month + "/" +
RIGHT(STR(report_year),2))
```

```
ptitle = "MORBIDITY REPORT FOR " + UPPER(CMONTH(report_date))+
" " +STR(YEAR( DATE()),4)
```

DO WHILE .NOT. EOF()

```
IF MONTH(physical) = MONTH(report_date) .AND.
```

```

YEAR(physical) = YEAR(report_date)
  physical_count = physical_count + 1
ENDIF

IF DELETED() = .T.
  checkout_count = checkout_count + 1
ENDIF
* Checkout dates are not presently stored if they are in
the future
* use the checkout count data the same as the other
counters.

IF MONTH(checkin) = MONTH(report_date) .AND.
YEAR(checkin) = YEAR(report_date)
  checkin_count = checkin_count + 1
ENDIF

IF MONTH(yellowfev) = MONTH(report_date) .AND.
YEAR(yellowfev) = YEAR(report_date)
  yf_count = yf_count + 1
ENDIF

IF MONTH(PPD) = MONTH(report_date) .AND.
YEAR(PPD) = YEAR(report_date)
  ppd_count = ppd_count + 1
ENDIF

IF MONTH(tetanus) = MONTH(report_date) .AND.
YEAR(tetanus) = YEAR(report_date)
  td_count = td_count + 1
ENDIF

IF MONTH(typhoid) = MONTH(report_date) .AND.
YEAR(typhoid) = YEAR(report_date)
  typ_count = typ_count + 1
ENDIF
SKIP
ENDDO
* Counts each field listed for the number completed in the
month given.
* Continues until EOF is retrieved.

@ 2,20 SAY ptitie
@ 6,8 SAY "Physical Exams"
@ 6,36 SAY "Checkins"
@ 6,56 SAY "Checkouts"
@ 7,13 SAY physical_count PICTURE "999"
@ 7,38 SAY checkin_count PICTURE "999"
@ 7,59 SAY checkout_count PICTURE "999"
@ 10,3 SAY "Yellowfever"
@ 10,28 SAY "PPD"

```



```

● 10,45 SAY "Tetanus"
● 10,64 SAY "Typhoid"
● 11,7 SAY yf_count PICTURE "999"
● 11,28 SAY ppd_count PICTURE "999"
● 11,47 SAY td_count PICTURE "999"
● 11,66 SAY typ_count PICTURE "999"
READ
CLEAR
* Puts results of the morbidity counts to the screen.

● 5,18 SAY "The month of this report is " +
    CMONTH(report_date)
● 8,14 SAY "Do you want this report sent to the printer"
● 10,35 GET print_job PICTURE "@M N,Y";
    MESSAGE "Press spacebar to toggle, ENTER to select."
READ
CLEAR
* Query to determine if a printed copy is needed.

IF print_job = "Y"
    SET PRINTER ON
    PRINTJOB
    _wrap = .T.
    _alignment = "center"
    ?
    ? ptitle
    _alignment = "left"
    ?
    ?
    ?
    ?
    ? "Physical Exams" STYLE "U" AT 8, "Checkins" STYLE "U" AT
    34, "Checkouts" STYLE "U" AT 57
    ? physical_count AT 13 PICTURE "999"
    ?? checkin_count AT 36 PICTURE "999"
    ?? checkout_count AT 60 PICTURE "999"
    * Using the marked fields as the count
    ?
    ?
    ?
    ? "Yellowfever" STYLE "U" AT 3, "PPD" STYLE "U" AT 26,
    "Tetanus" STYLE "U" AT 45, "Typhoid" STYLE "U" AT 66
    ? yf_count AT 7 PICTURE "999"
    ?? ppd_count AT 26 PICTURE "999"
    ?? td_count AT 47 PICTURE "999"
    ?? typ_count AT 68 PICTURE "999"
    ENDPRINTJOB
ENDIF
* If print_job is Y then report is sent to the printer.
EJECT PAGE
SET ESCAPE ON

```

```

*PROGRAM NAME:  OUT_RPRT
*PURPOSE      :  This is a report which displays:
*      1.  Name
*      2.  Rank(title)
*      3.  SSN
*      4.  Service
*      5.  Command
*      6.  Check-out Date (estimated)
*      of records that are marked for deletion.
*      --Individuals are grouped by their command.
*WRITTEN BY   :  Kevin Albert Bianchi
*LAST CHANGED:  01 AUG 93

```

```

SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER command
* New_Data DBF is used with command multiple index.

```

```

GO TOP
SET PRINTER ON
* Record pointer to the beginning of the DBF.

```

```

PUBLIC mcommand, ptitle, mlineno, title, mplength, mpageno,
selection

```

```

* Variables Initialized
MCOMMAND = ""
_pageno = 1
ptitle = "PERSONNEL RECORDS CURRENTLY MARKED FOR DELETION"
mlineno = 0
mplength = 60
mpageno = 1
selection = SPACE(1)
* Variables initialized

```

```

@ 5,10 SAY "Do you want the report printed or put to the
screen?"

```

```

@ 7,17 SAY "Select 'P' for print or 'S' for screen."

```

```

@ 10,35 GET selection PICTURE "@M P,S";

```

```

MESSAGE "Press SPACEBAR to scroll and ENTER to select"

```

```

READ
CLEAR

```

```

IF selection = "P"
PRINTJOB
DO OUT_PgHd
* Calls the page header program.

```

```

DO WHILE .NOT. EOF()
MCOMMAND = COMMAND
mcount = 0

```

```

DO WHILE COMMAND = MCOMMAND
  IF mlength = mlineno
    mlineno = 0
    EJECT PAGE
    DO OUT_PgHd
    DO OUT_Cmnd
  ENDIF
  * Line # counter to zero, page advanced, Header
  programs called.

  IF DELETED() = .T.
    mcount = mcount + 1
    IF mcount = 1
      DO OUT_Cmnd
    ENDIF
    * Prints command header when a new command is
    encountered.

    DO TITLE
    DO OUT_DATA
    * Output programs called
  ENDIF
  SKIP
ENDDO
* Continue while the command remains the same.
ENDDO
* Continues until EOF marker is located.
READ
EJECT PAGE
ENDPRINTJOB
ELSE
DO PRN_SCRN
* Procedure to prepare print variables for a put to screen.
PRINTJOB
DO OUT_Schd
* Calls the page header program.

DO WHILE .NOT. EOF()
  IF _plength = mlineno
    mlineno = 0
    EJECT PAGE
    DO OUT_Schd
  ENDIF
  * Line # counter to zero, page advanced, Header programs
  called.

  IF DELETED() = .T.
    DO TITLE
    DO OUT_DATA
    * Output programs called
  ENDIF

```

```

        SKIP
    ENDDO
    * Continues until EOF marker is located.
    READ
    ENDPRINTJOB
    DO PRN_RTRN
    * Procedure to return print variables to their original
    values.
ENDIF
SET ESCAPE ON

```

```

*PROGRAM NAME:  OUT_NEXT
*PURPOSE       :  This is a report which displays:
*               1.  Name
*               2.  Rank(title)
*               3.  SSN
*               4.  Service
*               5.  Command
*               6.  Check-out Date (estimated)
*               of personnel in the Medical ADMIN Unit's care who
*               are due to check-out in a given month.  The
*               program will query the individual for the month.
*               --Individuals are grouped by their command.
*WRITTEN BY    :  Kevin Albert Bianchi
*LAST CHANGED:  9 JUL 93

```

```

SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER command
* Uses new_data DBF and the command multiple index

```

```

GO TOP
SET PRINTER ON
*Record pointer to the top of the DBF printer set on.

```

```

PUBLIC mcommand, mlineno, checkout_month, checkout_year,
checkout_date,; ptitle, title, mplength, mpageno, selection
* variables declared global

```

```

* Variables initialized
MCOMMAND = ""
mlineno = 0
mpageno = 1
mplength = 60
print_flag = .T.
checkout_month = SPACE(2)
checkout_year = 1993
selection = SPACE(1)
* Variables initialized

```

```

@ 5,18 SAY "This report will list the personnel who are"
@ 6,18 SAY "due to checkout in the month you select."
@ 10,5 SAY "Enter the number equal to the month of your report
and press return."
@ 18,35 GET checkout_month PICTURE "@M
01,02,03,04,05,06,07,08,09,10,11,12";
MESSAGE "Press SPACEBAR to scroll, Enter to select."

```

```

READ
CLEAR

```

* Multiple choice query for report month.

```

@ 5,10 SAY "Enter the year which contains the month of this
report."
@ 7,11 SAY "***Enter all four numbers (I.E. For 1993 ENTER
1993)."
@ 12,35 GET checkout_year PICTURE "9999"

```

```

READ
CLEAR

```

* Query for year of report.

```

checkout_date = CTOD("01/" + checkout_month + "/" +
RIGHT(STR(checkout_year),2))
ptitle = "PERSONNEL DUE TO CHECKOUT IN " +
UPPER(CMONTH(checkout_date))

```

```

@ 5,10 SAY "Do you want the report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."
@ 10,35 GET selection PICTURE "@M P,S";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"

```

```

READ
CLEAR

```

* Assigns a value to selection which determines a print or a put to screen.

```

IF selection = "P"
PRINTJOB
DO OUT_PgHd
* Report page header

```

```

DO WHILE .NOT. EOF()
MCOMMAND = COMMAND

```

* Command name assigned to mcommand as a mem_var.

```

mcount = 0
DO WHILE COMMAND = MCOMMAND
IF mlineno = mlength
mlineno = 0
EJECT PAGE
DO OUT_PgHd

```

```

        DO OUT_Cmnd
    ENDIF
    * Line # counter to zero, full page is advanced,
    header programs called.

    IF VAL(checkout_month) >= MONTH(checkout)
    .AND. checkout_year = YEAR(checkout);
    .OR. checkout_year > YEAR(checkout)
    .AND. checkout <> {}
        mcount = mcount + 1
        IF mcount = 1
            DO OUT_Cmnd
        ENDIF
        * Determines if new command and calls the command
        header program.
        DO TITLE
        DO OUT_DATA
        * Title and Out_data called for output.
    ENDIF
    * Determines if check-out is due.
    SKIP
    ENDDO
    * Continues until command changes.
    ENDDO
    * Continues until EOF is retrieved.
    EJECT PAGE
    ENDPRINTJOB
ELSE
    DO PRN_SCRN
    * Procedure to prepare print variables for a put to screen.

    PRINTJOB
    DO OUT_ScHd
    * Report page header

    DO WHILE .NOT. EOF()
        IF _plength = mlineno
            mlineno = 0
            EJECT PAGE
            DO OUT_ScHd
        ENDIF
        * Line # counter to zero, full page is advanced, header
        programs called.

        IF VAL(checkout_month) >= MONTH(checkout) .AND.
        checkout_year = YEAR(checkout);
        .OR. checkout_year > YEAR(checkout) .AND. checkout <> {}
            DO TITLE
            DO OUT_DATA
            * Title and Out_data called for output.

```

```

        ENDIF
        * Determines if check-out is due.
        SKIP
    ENDDO
    * Continues until EOF is retrieved.
    ENDPRINTJOB
    DO PRN_RTRN
    * Procedure to return print variables to their original
    values.
ENDIF
SET ESCAPE ON

```

```

*PROGRAM NAME:  OUT_NOW
*PURPOSE      :  This is a report which displays:
*              1.  Name
*              2.  Rank(title)
*              3.  SSN
*              4.  Service
*              5.  Command
*              6.  Check-out Date (estimated)
*              of personnel in the Medical ADMIN Unit's care due
*              to check-out.
*              --Individuals are grouped by their command.
*WRITTEN BY   :  Kevin Albert Bianchi
*LAST CHANGED:  9 JUL 93

```

```

SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER command
* New_Data DBF is used with command multiple index.

```

```

GO TOP
SET PRINTER ON
* Record pointer to the beginning of the DBF.

```

```

PUBLIC mcommand, ptitle, mlineno, title, mplength, mpageno,
selection

```

```

* Variables Initialized
MCOMMAND = ""
_pageno = 1
ptitle = "PERSONNEL CURRENTLY LISTED TO CHECKOUT"
mlineno = 0
mplength = 60
mpageno = 1
selection = SPACE(1)
* Variables initialized

```

```

@ 5,10 SAY "Do you want the report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."
@ 10,35 GET selection PICTURE "@M P,S";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR
* Assigns a value to selection which determines a print or a
put to screen.

IF selection = "P"
PRINTJOB
DO OUT_PgHd
* Calls the page header program.

DO WHILE .NOT. EOF()
MCOMMAND = COMMAND
mcount = 0
DO WHILE COMMAND = MCOMMAND
IF mlength = mlineno
mlineno = 0
EJECT PAGE
DO OUT_PgHd
DO OUT_Cmnd
ENDIF
* Line # counter to zero, page advanced, Header
programs called.
IF checkout <= DATE()
count = mcount + 1
IF mcount = 1
DO OUT_Cmnd
ENDIF
* Prints command header when a new command is
encountered.
DO TITLE
DO OUT_DATA
* Output programs called
ENDIF
SKIP
ENDDO
* Continue while the command remains the same.
ENDDO
* Continues until EOF marker is located.
EJECT PAGE
ENDPRINTJOB
ELSE
DO PRN_SCRN
* Procedure to prepare print variables for a put to the
screen.

PRINTJOB

```



```

DO OUT_Schd
* Calls the page header program
DO WHILE .NOT. EOF()
    IF _plength = mlineno
        mlineno = 0
        EJECT PAGE
        DO OUT_PgHd
        DO OUT_Cmnd
    ENDIF
    * Line # counter to zero, page advanced, Header programs
    called.
    IF checkout <= DATE()
        DO TITLE
        DO OUT_DATA
        * Output programs called
    ENDIF
SKIP
ENDDO
* Continues until EOF marker is located.
ENDPRINTJOB
DO PRN_RTRN
* procedure to return print variables to their original
values.
ENDIF
SET ESCAPE ON

```

```

*PROGRAM NAME:   PFT_REP (Personnel Lacking Current Physical
                  Exam for the PFT)
*PURPOSE        :   This is a report which displays:
*                1.   Name
*                2.   Rank(title)
*                3.   SSN
*                4.   Date of last Physical Exam
*                5.   Date next Physical Exam is due
*                of personnel in the Medical ADMIN Unit's care.
*                --Individuals are grouped by their command.
*WRITTEN BY    :   Kevin Albert Bianchi
*LAST CHANGED:   6 Jul 93

```

```

SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER command
* New_data DBF used with the command multiple index.

```

```

GO TOP
* Record pointer to beginning of the DBF.

```

```

SET PRINTER ON

```

```

PUBLIC mlineno, snext_PE, mcommand, due, mplength, title,
selection
* Declare variables to be global

* Variables initialized
mcommand = ""
snext_PE = {}
due = .T.
ptitle = "PERSONNEL LACKING CURRENT PHYSICAL EXAM FOR THE PFT"
_pageno = 1
print_flag = .T.
mlineno = 0
mplength = 60
mpageno = 1
selection = SPACE(1)
* variables initialized

@ 5,10 SAY "Do you want the report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."
@ 10,35 GET selection PICTURE "@M P,S";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR
* Assigns a value to selection which determines a print or a
put to screen.

IF selection = "P"
PRINTJOB
DO PFT_PgHd
DO WHILE .NOT. EOF()
MCOMMAND = COMMAND
mcount = 0
DO WHILE COMMAND = MCOMMAND
DO PHYS_DUE
* Physical due date determined in called program.
DO PE_DUE_1
* Compares physical due date to the current date,
* due is defined logical true or false.
IF mplength = mlineno
EJECT PAGE
mlineno = 0
DO PFT_PgHd
DO PFT_Cmnd
ENDIF
* Full page advanced, line # to zero, command header
programs called.
IF DUE = .T.
mcount = mcount + 1
IF mcount = 1
DO PFT_Cmnd

```

```

        ENDIF
        * Determines if new command is started.
        DO TITLE
        IF physical = {12/12/12}
            DO DUE_DATA
        ELSE
            DO PFT_DATA
        ENDIF
        * For those records given 12/12/12 because they
        were due.
        * This was a result of initial data collection, as
        of the date
        * of this program all record reviews will
        ascertain the date of
        * the previous physical rather than just a due
        statement.
        SKIP
    ELSE
        SKIP
    ENDIF
    * If due is true report data is printed else, skip to
    next record.
ENDDO
ENDDO
EJECT PAGE
ENDPRINTJOB
ELSE
    DO PRN_SCRN
    * Procedure to prepare print variables for a put to screen.
    PRINTJOB
    DO PFT_Schd
    DO WHILE .NOT. EOF()
    DO PHYS_DUE
    * Physical due date determined in called program.
    DO PE_DUE_1
    * Compares physical due date to the current date,
    * due is defined logical true or false.
    IF _plength = mlineno
        EJECT PAGE
        mlineno = 0
        DO PFT_Schd
    ENDIF
    * Full page advanced, line # to zero, command header
    programs called.
    IF DUE = .T.
        DO TITLE
        IF physical = {12/12/12}
            DO DUE_DATA
        ELSE
            DO PFT_DATA
        ENDIF
    
```

```

* For those records given 12/12/12 because they were due.
* This was a result of initial data collection, as of the
date of this program all record reviews will ascertain the
date of the previous physical rather than just a due
statement.
    SKIP
ELSE
    SKIP
ENDIF
* If due is true report data is printed else, skip to next
record.
ENDDO
ENDPRINTJOB
DO PRN_RTRN
* Procedure to return print variables to their original
values.
ENDIF
SET ESCAPE ON

```

```

*PROGRAM NAME:  PFT_REP2 (Personnel Lacking Current Physical
                  Exam for the PFT)
*PURPOSE       :  This is a report which displays:
*               1.  Name
*               2.  Rank(title)
*               3.  SSN
*               4.  Date of last Physical Exam
*               5.  Date next Physical Exam is due
*               of personnel in the Medical ADMIN Unit's care who
                  will need
*               a physical exam before the PFT's scheduled date.
*               --Individuals are grouped by their command.
*WRITTEN BY    :  Kevin Albert Bianchi
*LAST CHANGED:  6 Jul 93

SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER command
* New_data DBF used with the command multiple index.

```

```

GO TOP
* Record pointer to beginning of the DBF.

```

```

SET PRINTER ON
PUBLIC mlineno, snext_PE, mcommand, due, ptitle, PFT_year,
month_val,; mplength, mpageno, title, selection
* Declare variables to be global

```

```

* Variables initialized
mcommand = ""

```

```

snext_PE = {}
due = .T.
PFT_month = SPACE (9)
PFT_year = 0
month_val = 0
mlinenno = 0
mplength = 60
mpageno = 1
selection = SPACE(1)
* variables initialized

```

```

@ 5,11 SAY "Select the month in which the PFT is going to be
performed?"

```

```

@ 7,35 GET PFT_month PICTURE "@M January, February, March,
April, May, June, July, August, September, October,;
November, December";

```

```

MESSAGE "Press SPACEBAR to scroll, ENTER to select"

```

```

READ

```

```

CLEAR

```

```

* Query month of report.

```

```

DO CASE

```

```

CASE PFT_month = "January"

```

```

month_val = 1

```

```

CASE PFT_month = "February"

```

```

month_val = 2

```

```

CASE PFT_month = "March"

```

```

month_val = 3

```

```

CASE PFT_month = "April"

```

```

month_val = 4

```

```

CASE PFT_month = "May"

```

```

month_val = 5

```

```

CASE PFT_month = "June"

```

```

month_val = 6

```

```

CASE PFT_month = "July"

```

```

month_val = 7

```

```

CASE PFT_month = "August"

```

```

month_val = 8

```

```

CASE PFT_month = "September"

```

```

month_val = 9

```

```

CASE PFT_month = "October"

```

```

month_val = 10

```

```

CASE PFT_month = "November"

```

```

month_val = 11

```

```

CASE PFT_month = "December"

```

```

month_val = 12

```

```

ENDCASE

```

```

* Case gets a numeric value for PFT-month and call it
month_val

```

```

DO WHILE PFT_year < YEAR( DATE( ) )

```

```

@ 5,17 SAY "What year is the PFT going to be performed?"
@ 6,22 SAY "Enter all four numbers of the year."
@ 9,35 GET PFT_year PICTURE "9999"
READ
CLEAR
IF PFT_year < YEAR(DATE())
    @ 5,6 SAY "Invalid value for year, value can not be less
        than the current year."
    READ
    CLEAR
ENDIF
* If the given value for PFT_year is too small loops back
for a reentry.
ENDDO
* Query for the year PFT will be performed.

ptitle = "PERSONNEL NEEDING A PHYSICAL EXAM FOR THE PFT IN "
        + UPPER(PFT_month)

@ 5,10 SAY "Do you want the report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."
@ 10,35 GET selection PICTURE "@M P,S";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR
* Assigns a value to selection which determines a print or a
put to screen.

IF selection = "P"
    PRINTJOB
    DO PFT_PgHd
    DO WHILE .NOT. EOF()
        MCOMMAND = COMMAND
        mcount = 0
        DO WHILE COMMAND = MCOMMAND
            DO PHYS_DUE
                * Physical due date determined in called program.
            DO PE_DUE_2
                * Compares physical due date to the month_val date,
                * due is defined logical true or false.

            IF mlength = mlineno
                EJECT PAGE
                mlineno = 0
                DO PFT_PgHd
                DO PFT_Cmnd
            ENDIF
            * Full page advanced, line # to zero, command header
            programs called.
            IF DUE = .T.

```

```

        mcount = mcount + 1
    IF mcount = 1
        DO PFT_Cmnd
    ENDIF
    * Determines if new command is started.
    DO TITLE
    IF physical = {12/12/12}
        DO DUE_DATA
    ELSE
        DO PFT_DATA
    ENDIF
    * This is for data entry value 12/12/12 which was
    used for due
    * when physical exam preformed dates were not
    ascertained during record review.
    * As of the date of this program further record
    reviews will determine the data the physical exams
    were performed.
    SKIP
ELSE
    SKIP
ENDIF
* If due is true report data is printed else, skip to
next record.
ENDDO
ENDDO
EJECT PAGE
ENDPRINTJOB
ELSE
    DO PRN_SCRN
    * Procedure to prepare print variables for a put to the
    screen.
    PRINTJOB
    DO PFT_ScHd
    DO WHILE .NOT. EOF()
        DO PHYS_DUE
        * Physical due date determined in called program.
        DO PE_DUE_2
        * Compares physical due date to the month_val date,
        * due is defined logical true or false.
        IF _plength = mlineno
            EJECT PAGE
            mlineno = 0
            DO PFT_ScHd
        ENDIF
        * Full page advanced, line # to zero, screen page
        header.

        IF DUE = .T.
            DO TITLE
            IF physical = {12/12/12}

```

```

        DO DUE_DATA
    ELSE
        DO PFT_DATA
    ENDIF
    * This is for data entry value 12/12/12 which was
    used for due when physical exam preformed dates were
    not ascertained during record review.
    * As of the date of this program further record
    reviews will determine the data the physical exams
    were performed.
    SKIP
ELSE
    SKIP
ENDIF
* If due is true report data is printed else, skip to
next record.
ENDDO
ENDPRINTJOB
DO PRN_RTRN
    * Procedure to return print variables to their original
    values.
ENDIF
SET ESCAPE ON

```

```

*PROGRAM NAME:   PFT_REP3 (Personnel Lacking Current Physical
                  Exam for the PFT)
*PURPOSE       :   This is a report which displays:
*               1.   Name
*               2.   Rank(title)
*               3.   SSN
*               4.   Date of last Physical Exam
*               5.   Date next Physical Exam is due
*               of personnel in the Medical ADMIN Unit's care
*               who will need a physical exam before the PFT's
*               scheduled date with a query by curriculum.
*               --Individuals are grouped by their command.
*WRITTEN BY    :   Kevin Albert Bianchi
*LAST CHANGED:  6 Jul 93

```

```

SET ESCAPE OFF
SET TALK OFF
USE new_data.dbf ORDER curric
* PFT_cur view used with the curric multiple index.

```

```

GO TOP
* Record pointer to beginning of the DBF.

```

```

SET PRINTER ON

```



```

PUBLIC mlineo, snext_PE, mcommand, due, ptitle, PFT_year,;
      PFT_month, month_val, mplength, mpageno, curric_code,;
      good_report, title, curric_name, selection
* Declare variables to be global

* Variables initialized
mcommand = ""
snext_PE = {}
due = .T.
PFT_month = SPACE (9)
PFT_year = 0
month_val = 0
mlineo = 0
mplength = 60
mpageno = 1
curric_code = SPACE(2)
good_report = "N"
selection = SPACE(1)
* variables initialized

DO WHILE UPPER(good_report) <> "Y"
  @ 5,6 SAY "Select the curriculum or department code
performing the PFT?"
  @ 7,16 SAY "The final option, 04, is for NPGS staff."
  @ 9,35 GET curric_code PICTURE "@M
30,31,32,33,3A,34,35,36,37,38,39,04";
  MESSAGE "Press SPACEBAR to scroll, ENTER to select"
  READ
  DO CODE_NAM
    * Retrieves curriculum office name for the given curriculum
code.
    @ 12,12 SAY "The curriculum you chose is " + curric_name
    @ 14,13 SAY "Is this the curriculum you want a report for?"
    @ 15,20 SAY "Select 'Y' for yes or 'N' for no."
    @ 17,35 GET good_report PICTURE "@M Y,N"
    MESSAGE "Press SPACEBAR to scroll and ENTER to select."
    READ
    CLEAR
  ENDDO

  @ 5,10 SAY "Select the month in which the PFT is going to be
performed?"
  @ 7,35 GET PFT_month PICTURE "@M January, February, March,
April, May,; June, July, August, September, October,;
November, December";
  MESSAGE "Press SPACEBAR to scroll, ENTER to select"
  READ
  CLEAR
  * Query month of report.

DO CASE

```

```

CASE PFT_month = "January"
    month_val = 1
CASE PFT_month = "February"
    month_val = 2
CASE PFT_month = "March"
    month_val = 3
CASE PFT_month = "April"
    month_val = 4
CASE PFT_month = "May"
    month_val = 5
CASE PFT_month = "June"
    month_val = 6
CASE PFT_month = "July"
    month_val = 7
CASE PFT_month = "August"
    month_val = 8
CASE PFT_month = "September"
    month_val = 9
CASE PFT_month = "October"
    month_val = 10
CASE PFT_month = "November"
    month_val = 11
CASE PFT_month = "December"
    month_val = 12
ENDCASE
* Case gets a numeric value for PFT-month and calls it
month_val

DO WHILE PFT_year < YEAR( DATE() )
    @ 5,17 SAY "What year is the PFT going to be performed?"
    @ 6,22 SAY "Enter all four numbers of the year."
    @ 9,35 GET PFT_year PICTURE "9999"
    READ
    CLEAR
    IF PFT_year < YEAR( DATE() )
        @ 5,6 SAY "Invalid value for year, value can not be less
        than the current year."
        READ
        CLEAR
    ENDIF
    * If the given value for PFT_year is too small loops back
    for a reentry.
ENDDO
* Query for the year PFT will be performed.

ptitle = "PHYSICAL EXAM DUE FOR THE PFT IN "+UPPER(PFT_month);
        + "(" + UPPER(curric_name) + ")"

@ 5,10 SAY "Do you want the report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."

```

```

@ 10,35 GET selection PICTURE "@M P,S";
      MESSAGE "Press SPACEBAR to scroll and ENTER to select."
READ
CLEAR
* Assigns a value to selection which determines a print or a
put to screen.

IF selection = "P"
  PRINTJOB
  DO PFT_PgHd
  DO WHILE .NOT. EOF()
    DO WHILE curric_div = curric_code
      DO PHYS_DUE
        * Physical due date determined in called program.
      DO PE_DUE_2
        * Compares physical due date to the month_val date,
        * due is defined logical true or false.
      IF mlineno = mplength
        EJECT PAGE
        mlineno = 0
        DO PFT_PgHd
        DO PFT_Cmnd
      ENDIF
      * Full page advanced, line # to zero, command header
      programs called.
      IF DUE = .T.
        DO TITLE
        IF physical = {12/12/12}
          DO DUE_DATA
        ELSE
          DO PFT_DATA
        ENDIF
        * This is for the 12/12/12 data entry for physical
        which is the entry for due.
        * In the future this field will have the actual
        * date the physical exam was completed.
      ENDIF
      * If due is true report data is printed else, skip to
      next record.
      SKIP
    ENDDO
  SKIP
  ENDDO
  READ
  EJECT PAGE
  ENDPRINTJOB
ELSE
  DO PRN_SCRN
  * Procedure to prepare print variables for a put to the
  screen.
  PRINTJOB

```

```

DO PFT_Schd
DO WHILE .NOT. EOF()
  DO WHILE curric_div = curric_code
    DO PHYS_DUE
      * Physical due date determined in called program.
    DO PE_DUE_2
      * Compares physical due date to the month_val date,
      * due is defined logical true or false.

    IF _plength = mlineno
      EJECT PAGE
      mlineno = 0
      DO PFT_Schd
    ENDIF
    * Full page advanced, line # to zero, screen header
    programs called.
    IF DUE = .T.
      DO TITLE
      IF physical = {12/12/12}
        DO DUE_DATA
      ELSE
        DO PFT_DATA
      ENDIF
      * This is for the 12/12/12 data entry for physical
      which is the
      * entry for due. In the future this field will
      have the actual
      * date the physical exam was completed.
    ENDIF
    * If due is true report data is printed else, skip to
    next record.
    SKIP
  ENDDO
  SKIP
ENDDO
READ
ENDPRINTJOB
DO PRN_RTRN
  * Procedure to return print variables to their original
  values.
ENDIF
SET ESCAPE OFF

```

```

*PROGRAM NAME:  ONE_STOP
*PURPOSE      :  This is a report which displays:
*              1.  Name
*              2.  Rank(title)
*              3.  SSN
*              4.  Medical Action Required
*              of personnel who are due for at least one of the
*              following immunizations, physical exam and blood
*              type.
*              The program will query for the major commands.
*              This is designed to support on stop check-out at
*              NPGS but will run for MCD and NSGD in case they
*              establish a one stop check-out.
*WRITTEN BY   :  Kevin Albert Bianchi
*LAST CHANGED:  20 AUG 93

```

```

SET TALK OFF
SET ESCAPE OFF
SET CENTURY ON
* Allows for 21ST century values in dates.
USE New Data ORDER command
* use the new_data DBF with the command multiple index

```

```

GO TOP
* Set record pointer at the beginning of the DBF.
SET PRINTER ON

```

```

PUBLIC report_date, wrong_input next_PE, next_PPD, next_YF,
      next_TD,; next_TYP, ptitle, mprint, snext_PE, title,
      act_phys,; act_blood, act_G6PD, act_sikcel, act_ppd,
      act_yf, act_td, act_typ,; mlineno, good_report,
      report_day, report_month, report_year,; mpageno, mplength,
      selection, command_report
* Declares variables global.

```

```

* Variable Initialized
_pageno = 1
mprint = .T.
report_month = SPACE(2)
report_day = 0
report_year = 0
wrong_input = 0
report_date = {}
good_report = SPACE(1)
next_PE = {}
next_PPD = {}
next_YF = {}
next_TD = {}
next_TYP = {}
mlineno = 0
mplength = 60

```

```

mpageno = 1
command_report = SPACE(4)
selection = SPACE(1)
* Variables initialized

DO WHILE UPPER(good_report) <> "Y"
  @ 5,10 SAY "This report will list the personnel who are due
  for medical"
  @ 6,14 SAY "action prior to the date you enter on this
  screen."
  @ 7,10 SAY "It will list only those who are marked for
  deletion from the"
  @ 8,18 SAY "command you select later in this program."
  DO WHILE wrong_input < 1 .OR. wrong_input > 31
    @ 10,20 SAY "Enter the day of the month of this report."
    @ 12,6 SAY "***Enter the leading zero if value is 1-9
    (I.E. For 1 Jan ENTER 01)."
    @ 15,35 get report_day PICTURE "@Z 99"
    READ
    * Query the user for the cut off day of this report.

    wrong_input = report_day
    IF wrong_input < 1 .OR. wrong_input > 31
      @ 17,5 SAY "Incorrect input, the number entered must
      be a value between 1 and 31."
      @ 18,25 SAY "Press ENTER and try again."
      READ
    ENDIF
    * message put to the screen if the value entered for day
    is invalid
    CLEAR
  ENDDO
  * Loops until proper day value is input.

  @ 10,4 SAY "Select the number equal to the month of your
  report and press return."
  @ 18,35 GET report_month PICTURE "@M
  01,02,03,04,05,06,07,08,09,10,11,12";
  MESSAGE "Press SPACEBAR to scroll, ENTER to select."
  READ
  CLEAR
  * Query the user for the first month of the report
  (multiple choice).

  wrong_input = 0

  DO WHILE wrong_input < YEAR(DATE())
    @ 5,10 SAY "Enter the year which contains the month of
    this report."
    @ 7,11 SAY "***Enter all four numbers (I.E. For 1993
    ENTER 1993)."

```

```

@ 12,35 get report_year PICTURE "@Z 9999"
READ
* Query the user for the year of the first month of the
report.

wrong_input = report_year
IF wrong_input < YEAR(DATE())
    @ 15,18 SAY "Incorrect input, the number entered must
    be"
    @ 16,18 SAY "a value greater than the current year's
    date."
    @ 18,25 SAY "Press ENTER and try again."
    READ
ENDIF
CLEAR
* Message put to screen if the value for the year is too
small.
ENDDO
* Loops until proper year data is input.

report_date = CTOD(RIGHT(STR(report_day),2) + "/" +
report_month + "/" + RIGHT(STR(report_year),2))
* report_date is the date of the report

@ 5,18 SAY "The date of the report you selected is:"
@ 7,32 SAY DTOC(report_date)
@ 9,13 SAY "If this is correct select 'Y' (for yes) to
continue."
@ 10,7 SAY "If not select 'N' (for no) to re-initiate the
date of this report."
@ 12,35 GET good_report PICTURE "@M Y,N";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR
* Final query to ensure the proper date was input prior to
printing report.
ENDDO

@ 5,6 SAY "This report displays readiness data for records
marked for deletion."
@ 7,15 SAY "Select the command for which this report is
needed."
@ 10,34 GET command_report PICTURE "@M NPGS,NSGD,MCD";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR

ptitle = "ONE STOP CHECK-OUT MEDICAL READINESS REPORT FOR " +
UPPER(command_report)

```

* Title declared after variable command_report assigned a value.

@ 5,10 SAY "Do you want this report printed or put to the screen?"

@ 7,17 SAY "Select 'P' for print or 'S' for screen."

@ 10,35 GET selection PICTURE "@M P,S";

MESSAGE "Press SPACEBAR to scroll and ENTER to select"

READ

CLEAR

* Assigns a value to selection which determines a print or a put to screen.

IF selection = "P"

PRINTJOB

DO RED_PgHd

* Calls page header program

DO WHILE .NOT. EOF()

IF command = command_report

* Locates the records of the selected commands.

DO WHILE command = command_report

* Only the selected command's records searched for deletion mark.

IF mlineno = mlength

mlineno = 0

EJECT PAGE

DO RED_PgHd

ENDIF

* Line # count to zero, page advanced, and page header program called.

IF DELETED() = .T.

DO PHYS_DUE

IF snext_PE <= report_date

act_phys = "DUE"

ELSE

act_phys = ""

ENDIF

* Determines if physical is due during the date of this report.

IF UPPER(blood) = "DUE"

act_blood = "DUE"

ELSE

act_blood = ""

ENDIF

* Determines if blood type is unknown.

IF UPPER(G6PD) = "DUE"

act_G6PD = "DUE"

ELSE


```

    act_G6PD = ""
ENDIF
* Determines if G6PD is due during the date of
this report.

IF UPPER(sicklecell) = "D"
    act_SC = "DUE"
ELSE
    act_SC = ""
ENDIF
* Determines if sickle cell is due during the
date of this report.

DO PPD_DUE
IF next_PPD <= report_date
.AND. next_PPD > {01/01/1912}
* Lower limit of 01/01/1912 is due to the fact
* that 11/11/11 is a temporary code date for a
* allergic reaction and 12/12/12 is temporary
code date for immunization is due when data was
entered.
    act_PPD = "DUE"
ELSE
    act_PPD = ""
ENDIF
* Determines if PPD is due during the date of
this report.

DO YF_DUE
IF next_YF <= report_date
.AND. next_YF > {01/01/1912}
    act_YF = "DUE"
ELSE
    act_YF = ""
ENDIF
* Determines if yellow fever is due during the
date of this report.

DO TD_DUE
IF next_TD <= report_date
.AND. next_TD > {01/01/1912}
    act_TD = "DUE"
ELSE
    act_TD = ""
ENDIF
* Determines if tetanus is due during the date
of this report.

DO TYP_DUE
IF next_TYP <= report_date
.AND. next_TYP > {01/01/1912};

```

```

.OR. BSC = .F.
    act_TYP = "DUE"
ELSE
    act_TYP = ""
ENDIF
* Determines if typhoid is due during the date
of this report.

IF act_phys = "DUE" .OR. act_blood = "DUE"
.OR. act_G6PD = "DUE";
.OR. act_SC = "DUE" .OR. act_PPD = "DUE"
.OR. act_YF = "DUE";
.OR. act_TD = "DUE" .OR. act_TYP = "DUE"
    DO TITLE
    DO RED_DATA
    * Prints readiness information for everyone
    marked for deletion who has at least one field
    requiring medical action.
ENDIF
SKIP
ENDDO
ENDIF
SKIP
ENDDO
* Loop until EOF marker is retrieved.
EJECT PAGE
ENDPRINTJOB
ELSE
DO PRN_SCRN
* Procedure to prepare print variables for a put to the
screen.
PRINTJOB
DO RED_Schd
* Calls the page header program for a put to screen
DO WHILE .NOT. EOF()
    IF command = command_report
    * Locates the records of the selected commands.
    DO WHILE command = command_report
    * Only the selected command's records searched for
    deletion mark.
        IF _plength = mlineno
            mlineno = 0
            EJECT PAGE
            DO RED_Schd
        ENDIF
    * Line # count to zero, page advanced, and page
    header program called.
    IF DELETED() = .T.
        DO PHYS_DUE
        IF snext_PE <= report_date
            act_phys = "DUE"

```

```

ELSE
    act_phys = ""
ENDIF
* Determines if physical is due during the date
of this report.

IF UPPER(blood) = "DUE"
    act_blood = "DUE"
ELSE
    act_blood = ""
ENDIF
* Determines if blood type is unknown.

IF UPPER(G6PD) = "DUE"
    act_G6PD = "DUE"
ELSE
    act_G6PD = ""
ENDIF
* Determines if G6PD is due during the date of
this report.

IF UPPER(sicklecell) = "D"
    act_SC = "DUE"
ELSE
    act_SC = ""
ENDIF
* Determines if sickle cell is due during the
date of this report.

DO PPD_DUE
IF next_PPD <= report_date
.AND. next_PPD > {01/01/1912}
* Lower limit of 01/01/1912 is due to the fact
that 11/11/11 is a temporary code date for
allergic reaction and 12/12/12 is a temporary
* code date for immunization is due when data
was entered.

    act_PPD = "DUE"
ELSE
    act_PPD = ""
ENDIF
* Determines if PPD is due during the date of
this report.

DO YF_DUE
IF next_YF <= report_date
.AND. next_YF > {01/01/1912};
.OR. BSC = .F.
    act_YF = "DUE"
ELSE

```

```

        act_YF = ""
    ENDIF
    * Determines if yellow fever is due during the
    date of this report.

    DO TD_DUE
    IF next_TD <= report_date
    .AND. next_TD > {01/01/1912}
        act_TD = "DUE"
    ELSE
        act_TD = ""
    ENDIF
    * Determines if tetanus is due during the date
    of this report.

    DO TYP_DUE
    IF next_TYP <= report_date
    .AND. next_TYP > {01/01/1912}
        act_TYP = "DUE"
    ELSE
        act_TYP = ""
    ENDIF
    * Determines if typhoid is due during the date
    of this report.

    IF act_phys = "DUE" .OR. act_blood = "DUE"
    .OR. act_G6PD = "DUE";
    .OR. act_SC = "DUE" .OR. act_PPD = "DUE"
    .OR. act_YF = "DUE"; .OR. act_TD = "DUE"
    .OR. act_TYP = "DUE"
        DO TITLE
        DO RED_DATA
    ENDIF
    * Prints readiness information for everyone
    marked for deletion who has at least one field
    requiring medical action.
    ENDIF
    SKIP
    ENDDO
    ENDIF
    SKIP
    ENDDO
    * Loop until EOF marker is retrieved.
    EJECT PAGE
    ENDPRINTJOB
ENDIF
SET ESCAPE ON
SET CENTURY OFF

```

```

*PROGRAM NAME:  READINES
*PURPOSE      :  This is a report which displays:
*              1.  Name
*              2.  Rank(title)
*              3.  SSN
*              4.  Medical Action Required
*              for personnel who are due for at least on of the
*              following immunizations, physical exam and blood type
*              --The program will query the individual for the period
*              of the report.
*              --Individuals are grouped by their command.
*WRITTEN BY   :  Kevin Albert Bianchi
*LAST CHANGED:  9 JUL 93

```

```

SET TALK OFF
SET ESCAPE OFF
SET CENTURY ON
* Allows for 21ST century values for year.
USE New_Data ORDER command
* use the new_data DBF with the command multiple index

```

```

GO TOP
* Set record pointer at the beginning of the DBF.
SET PRINTER ON

```

```

PUBLIC  begin_date,  final_date,  begin_date,  final_date,
        wrong_input,  next_PE,;  next_PPD,  next_YF,  next_TD,
        next_TYP,  ptitle,  mprint,  mcommand,;  snext_PE,  title,
        act_phys,  act_blood,  act_G6PD,  act_sikcel,  act_ppd,;
        act_yf,  act_td,  act_typ,  mlineno,  final_titledate,
        good_report,; mpageno, mplength, selection
* Declares variables global.

```

```

* Variable Initialized
_pageno = 1
mcommand = ""
mprint = .T.
begin_month = SPACE(2)
final_month = SPACE(2)
begin_year = 0
final_year = 0
wrong_input = 0
begin_date = {}
final_date = {}
final_titledate = {}
good_report = SPACE(1)
next_PE = {}
next_PPD = {}
next_YF = {}
next_TD = {}
next_TYP = {}

```

```

mlineno = 0
mplength = 60
mpageno = 1
selection = SPACE(1)
* Variables initialized

DO WHILE UPPER(good_report) <> "Y"
  @ 5,5 SAY "This report will list the personnel who are due
  for medical action."
  @ 10,1 SAY "Select the number equal to the first month of
  your report and press return."
  @ 18,35 GET begin_month PICTURE "@M
  01,02,03,04,05,06,07,08,09,10,11,12";
  MESSAGE "Press SPACEBAR to scroll, ENTER to select."
  READ
  CLEAR
  * Query the user for the first month of the report
  (multiple choice).

  wrong_input = 0

  DO WHILE wrong_input < YEAR(DATE())
    @ 5,10 SAY "Enter the year which contains the first
    month of this report."
    @ 7,14 SAY "Enter all four numbers (I.E. For 1993
    ENTER 1993)."
    @ 12,35 GET begin_year PICTURE "@Z 9999"
    READ
    * Query the user for the year of the first month of the
    report.

    wrong_input = begin_year
    IF wrong_input < YEAR(DATE())
      @ 15,18 SAY "Incorrect input, the number entered must
      be"
      @ 16,18 SAY "a value greater than the current years
      date."
      @ 18,25 SAY "Press ENTER and try again."
      READ
      CLEAR
    ENDIF
    CLEAR
    * Message put to screen if the value for the year is too
    small.
  ENDDO
  * Loops until proper year data is input.

  @ 10,1 SAY "Select the number equal to the final month of
  your report and press return."
  @ 18,35 GET final_month PICTURE "@M
  01,02,03,04,05,06,07,08,09,10,11,12";

```

```

    MESSAGE "Press SPACEBAR to scroll, ENTER to select."
READ
CLEAR
* Query the user for the year of the final month of the
report.

wrong_input = 0

DO WHILE wrong_input < YEAR(DATE())
    @ 5,10 SAY "Enter the year which contains the last month
of this report."
    @ 7,14 SAY "****Enter all four numbers (I.E. For 1993
ENTER 1993)."
    @ 12,35 get final_year PICTURE "@Z 9999"
    READ
    wrong_input = final_year
    IF wrong_input < YEAR(DATE())
        @ 15,18 SAY "Incorrect input, the number entered
must"
        @ 16,18 SAY "be a value greater than the current
year."
        @ 18,25 SAY "Press ENTER and try again."
    READ
    CLEAR
    ENDIF
    CLEAR
ENDDO
* Loops until proper year data is input.

begin_date = CTOD("01/" + begin_month + "/" +
RIGHT(STR(begin_year),2))
final_titledate = CTOD("28/" + final_month + "/" +
RIGHT(STR(final_year),2))
final_date = CTOD("01/" + STR(VAL(final_month)+1,2) + "/" +
+ RIGHT(STR(final_year),2))
ptitle = "MEDICAL READINESS REPORT FOR " +
UPPER(CMONTH(begin_date));
+ " THROUGH " + UPPER(CMONTH(final_titledate))
* begin_date is the initial date of the report
* final_titledate is the final date of the report used in
the title (the day value is not valid only the month is
used).
* final_date is the first day after the final date, used
for computations.

@ 5,17 SAY "The dates of the report you selected are:"
@ 7,24 SAY DTOC(begin_date) + " through to " +
DTOC(final_date)
@ 9,13 SAY "If this is correct select 'Y' (for yes) to
continue."

```

```

● 10,7 SAY "If not select 'N' (for no) to re-initiate the
date of this report."
● 12,35 GET good_report PICTURE "@M Y,N";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR
* Final query to ensure the proper date was input prior to
printing report.
ENDDO

@ 5,10 SAY "Do you want this report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."
@ 10,35 GET selection PICTURE "@M P,S";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR
* Assigns a value to selection which determines a print or a
put to screen.

IF selection = "P"
PRINTJOB
DO RED_PgHd
DO WHILE .NOT. EOF()
MCOMMAND = COMMAND
mcount = 0
DO WHILE COMMAND = MCOMMAND
IF mlength = mlength
mcount = 0
EJECT PAGE
DO RED_PgHd
DO RED_Cmdnd
ENDIF
* Full page is advanced, line # count to zero and
header programs called.
DO PHYS_DUE
IF snext_PE >= begin_date .AND. snext_PE < final_date
act_phys = "DUE"
ELSE
act_phys = ""
ENDIF
* Determines if physical is due during the date of
this report.

IF UPPER(blood) = "DUE"
act_blood = "DUE"
ELSE
act_blood = ""
ENDIF
* Determines if blood type is unknown.

```



```

IF UPPER(G6PD) = "DUE"
    act_G6PD = "DUE"
ELSE
    act_G6PD = ""
ENDIF
* Determines if G6PD is due during the date of this
report.

IF UPPER(sicklecell) = "D"
    act_SC = "DUE"
ELSE
    act_SC = ""
ENDIF
* Determines if sickle cell is due during the date of
this report.

DO PPD_DUE
IF next_PPD >= begin_date
    .AND. next_PPD < final_date;
    .OR. next_PPD = {12/12/1912}
    * The condition next_PPD = {12/12/1912} is a result
of a temporary
    * data entry code date in which 12/12/12 represents
the immunization
    * is due all future data entry will be the date
completed.
    act_PPD = "DUE"
ELSE
    act_PPD = ""
ENDIF
* Determines if PPD is due during the date of this
report.

DO YF_DUE
IF next_YF >= begin_date .AND. next_YF < final_date;
    .OR. next_YF = {12/12/1912}
    act_YF = "DUE"
ELSE
    act_YF = ""
ENDIF
* Determines if yellow fever is due during the date
of this report.

DO TD_DUE
IF next_TD >= begin_date .AND. next_TD < final_date;
    .OR. next_TD = {12/12/1912}
    act_TD = "DUE"
ELSE
    act_TD = ""
ENDIF

```

```

* Determines if tetanus is due during the date of
this report.

DO TYP_DUE
IF next_TYP >= begin_date
.AND. next_TYP < final_date;
.OR. BSC = .F. .OR. next_TYP = {12/12/1912}
    act_TYP = "DUE"
ELSE
    act_TYP = ""
ENDIF
* Determines if typhoid is due during the date of
this report.

IF act_phys = "DUE" .OR. act_blood = "DUE"
.OR. act_G6PD = "DUE";
.OR. act_SC = "DUE" .OR. act_PPD = "DUE"
.OR. act_YF = "DUE";
.OR. act_TD = "DUE" .OR. act_TYP = "DUE"
    mcount = mcount + 1
    IF mcount = 1
        DO RED_Cmdnd
    ENDIF
    * When mcount is 1 a new command header is
    printed.
    DO TITLE
    DO RED_DATA
ENDIF
* Prints readiness information for everyone who
* has at least one field requiring medical action.
SKIP
ENDDO
* Loop while command does not change.
ENDDO
* Loop until EOF marker is retrieved.
EJECT PAGE
ENDPRINTJOB
ELSE
DO PRN_SCRN
* Procedure to prepare print variables for a put to the
screen.
PRINTJOB
DO RED_Schd
DO WHILE .NOT. EOF()
    IF _plength = mlineno
        mlineno = 0
        EJECT PAGE
        DO RED_Schd
    ENDIF
    * Full page is advanced, line # count to zero and header
    programs called.

```

```

DO PHYS_DUE
IF snext_PE >= begin_date
.AND. snext_PE < final_date
    act_phys = "DUE"
ELSE
    act_phys = ""
ENDIF
* Determines if physical is due during the date of this
report.

IF UPPER(blood) = "DUE"
    act_blood = "DUE"
ELSE
    act_blood = ""
ENDIF
* Determines if blood type is unknown.
IF UPPER(G6PD) = "DUE"
    act_G6PD = "DUE"
ELSE
    act_G6PD = ""
ENDIF
* Determines if G6PD is due during the date of this
report.

IF UPPER(sicklecell) = "D"
    act_SC = "DUE"
ELSE
    act_SC = ""
ENDIF
* Determines if sickle cell is due during the date of
this report.

DO PPD_DUE
IF next_PPD >= begin_date .AND. next_PPD < final_date;
.OR. next_PPD = {12/12/1912}
* The condition next_PPD = {12/12/1912} is a result
of a temporary
* data entry code date in which 12/12/12 represents
the immunization
* is due all future data entry will be the date
completed.
    act_PPD = "DUE"
ELSE
    act_PPD = ""
ENDIF
* Determines if PPD is due during the date of this
report.

DO YF_DUE
IF next_YF >= begin_date .AND. next_YF < final_date;
.OR. next_YF = {12/12/1912}

```

```

        act_YF = "DUE"
ELSE
    act_YF = ""
ENDIF
* Determines if yellow fever is due during the date of
this report.

DO TD_DUE
IF next_TD >= begin_date .AND. next_TD < final_date;
.OR. next_TD = {12/12/1912}
    act_TD = "DUE"
ELSE
    act_TD = ""
ENDIF
* Determines if tetanus is due during the date of this
report.

DO TYP_DUE
IF next_TYP >= begin_date .AND. next_TYP < final_date;
.OR. BSC = .F. .OR. next_TYP = {12/12/1912}
    act_TYP = "DUE"
ELSE
    act_TYP = ""
ENDIF
* Determines if typhoid is due during the date of this
report.
IF act_phys = "DUE" .OR. act_blood = "DUE"
.OR. act_G6PD = "DUE"; .OR. act_SC = "DUE"
.OR. act_PPD = "DUE" .OR. act_YF = "DUE";
.OR. act_TD = "DUE" .OR. act_TYP = "DUE"
    DO TITLE
    DO RED_DATA
ENDIF
* Prints readiness information for everyone who
* has at least one field requiring medical action.
SKIP
ENDDO
* Loop until EOF marker is retrieved.
EJECT PAGE
ENDPRINTJOB
DO PRN_RTRN
* Procedure to return print variables to their original
values.
ENDIF
SET CENTURY OFF
SET ESCAPE ON

```

```

*PROGRAM NAME:  READINS2
*PURPOSE      :  This is a report which displays:
*              1.  Name
*              2.  Rank(title)
*              3.  SSN
*              4.  Medical Action Required
*              of personnel who are due for at least one of the
*              following immunizations, physical exam and blood type.
*              --The program will query the user for the date of the
*              report and all medical action due prior to that date
*              will be printed.
*              --Individuals are grouped by their command.
*WRITTEN BY   :  Kevin Albert Bianchi
*LAST CHANGED:  27 JUL 93

```

```

SET TALK OFF
SET ESCAPE OFF
SET CENTURY ON
* Allows for 21ST century values in dates.
USE New_Data ORDER command
* use the new_data DBF with the command multiple index

```

```

GO TOP
* Set record pointer at the beginning of the DBF.
SET PRINTER ON

```

```

PUBLIC report_date, wrong_input, next_PE, next_PPD, next_YF,
       next_TD,; next_TYP, ptitle, mprint, mcommand, snext_PE,
       title, act_phys,; act_blood, act_GGPD, act_sikcel, act_ppd,
       act_yf, act_td, act_typ,; mlineno, good_report, report_day,
       report_month, report_year,; mpageno, mplength, selection
* Declares variables global.

```

```

* Variable Initialized
_pageno = 1
mcommand = ""
mprint = .T.
report_month = SPACE(2)
report_day = 0
report_year = 0
wrong_input = 0
report_date = {}
good_report = "N"
next_PE = {}
next_PPD = {}
next_YF = {}
next_TD = {}
next_TYP = {}
mlineno = 0
mplength = 60
mpageno = 1

```

```

selection = SPACE(1)
* Variables initialized

DO WHILE UPPER(good_report) <> "Y"
  @ 5,10 SAY "This report will list the personnel who are due
  for medical"
  @ 6,12 SAY "action prior to the date you enter here on this
  screen."
  DO WHILE wrong_input < 1 .OR. wrong_input > 31
    @ 10,20 SAY "Enter the day of the month of this report."
    @ 12,6 SAY "Enter the leading zero if value is 1-9
    (I.E. For 1 Jan ENTER 01)."
    @ 15,35 get report_day PICTURE "@Z 99"
    READ
    * Query the user for the cut off day of this report.

    wrong_input = report_day
    IF wrong_input < 1 .OR. wrong_input > 31
      @ 17,5 SAY "Incorrect input, the number entered must
      be a value between 1 and 31."
      @ 18,25 SAY "Press ENTER and try again."
      READ
    ENDIF
    * message put to the screen if the value entered for day
    is invalid
    CLEAR
  ENDDO
  * Loops until proper day value is input.

  @ 10,4 SAY "Select the number equal to the month of your
  report and press return."
  @ 18,35 GET report_month PICTURE "@M
  01,02,03,04,05,06,07,08,09,10,11,12";
  MESSAGE "Press SPACEBAR to scroll, ENTER to select."
  READ
  CLEAR
  * Query the user for the month of the report (multiple
  choice).

  wrong_input = 0

  DO WHILE wrong_input < YEAR(DATE())
    @ 5,10 SAY "Enter the year which contains the month of
    this report."
    @ 7,11 SAY "Enter all four numbers (I.E. For 1993
    ENTER 1993)."
    @ 12,35 get report_year PICTURE "@Z 9999"
    READ
    * Query the user for the year of the first month of the
    report.

```

```

wrong_input = report_year
IF wrong_input < YEAR(DATE())
    @ 15,18 SAY "Incorrect input, the number entered must
    be"
    @ 16,18 SAY "a value greater than the current year's
    date."
    @ 18,25 SAY "Press ENTER and try again."
    READ
ENDIF
CLEAR
* Message put to screen if the value for the year is too
small.
ENDDO
* Loops until proper year data is input.

report_date = CTOD(RIGHT(STR(report_day),2) + "/" +
report_month + "/" + RIGHT(STR(report_year),2))

ptitle = "MEDICAL READINESS REPORT FOR " +
DTOC(report_date)
* report_date is the date of the report

@ 5,18 SAY "The date of the report you selected is:"
@ 7,32 SAY DTOC(report_date)
@ 9,13 SAY "If this is correct select 'Y' (for yes) to
continue."
@ 10,7 SAY "If not select 'N' (for no) to re-initiate the
date of this report."
@ 12,35 GET good_report PICTURE "@M Y,N";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR
* Final query to ensure the proper date was input prior to
printing report.
ENDDO

@ 5,10 SAY "Do you want this report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."
@ 10,35 GET selection PICTURE "@M P,S";
MESSAGE "Press SPACEBAR to scroll and ENTER to select"
READ
CLEAR
* Assigns a value to selection which determines a print or a
put to screen.

IF selection = "P"
    PRINTJOB
    DO RED_PgHd
    DO WHILE .NOT. EOF()

```

```

MCOMMAND = COMMAND
mcount = 0
DO WHILE COMMAND = MCOMMAND
  IF mlineno = mplength
    mlineno = 0
    EJECT PAGE
    DO RED_PgHd
    DO RED_Cmnd
      ENDIF
      * Full page is advanced, line # count to zero and
      header programs called.
      DO PHYS_DUE
      IF snext_PE <= report_date
        act_phys = "DUE"
      ELSE
        act_phys = ""
      ENDIF
      * Determines if physical is due during the date of
      this report.

      IF UPPER(blood) = "DUE"
        act_blood = "DUE"
      ELSE
        act_blood = ""
      ENDIF
      * Determines if blood type is unknown.

      IF UPPER(G6PD) = "DUE"
        act_G6PD = "DUE"
      ELSE
        act_G6PD = ""
      ENDIF
      * Determines if G6PD is due during the date of this
      report.

      IF UPPER(sicklecell) = "D"
        act_SC = "DUE"
      ELSE
        act_SC = ""
      ENDIF
      * Determines if sickle cell is due during the date of
      this report.

      DO PPD_DUE
      IF next_PPD <= report_date
        .AND. next_PPD > {01/01/1912}
        * Lower limit of 01/01/1912 is due to the fact that
        11/11/11 is a
        * temporary code date for allergic reaction and
        12/12/12 is a temporary

```



```

* code date for immunization is due when data was
entered.
    act_PPD = "DUE"
ELSE
    act_PPD = ""
ENDIF
* Determines if PPD is due during the date of this
report.

DO YF_DUE
IF next_YF <= report_date
.AND. next_YF > {01/01/1912}
    act_YF = "DUE"
ELSE
    act_YF = ""
ENDIF
* Determines if yellow fever is due during the date
of this report.

DO TD_DUE
IF next_TD <= report_date
.AND. next_TD > {01/01/1912}
    act_TD = "DUE"
ELSE
    act_TD = ""
ENDIF
* Determines if tetanus is due during the date of
this report.

DO TYP_DUE
IF next_TYP <= report_date
.AND. next_TYP > {01/01/1912} .OR. BSC = .F.
    act_TYP = "DUE"
ELSE
    act_TYP = ""
ENDIF
* Determines if typhoid is due during the date of
this report.

IF act_phys = "DUE" .OR. act_blood = "DUE"
.OR. act_G6PD = "DUE";
.OR. act_SC = "DUE" .OR. act_PPD = "DUE"
.OR. act_YF = "DUE";
.OR. act_TD = "DUE" .OR. act_TYP = "DUE"
    mcount = mcount + 1
    IF mcount = 1
        DO RED_Cmnd
    ENDIF
    * When mcount is 1 a new command header is
    printed.
    DO TITLE

```

```

        DO RED_DATA
        ENDIF
        * Prints readiness information for everyone who
        * has at least one field requiring medical action.
        SKIP
    ENDDO
    * Loop while command does not change.
ENDDO
* Loop until EOF marker is retrieved.
EJECT PAGE
ENDPRINTJOB
ELSE
DO PRN_SCRN
* Procedure to prepare print variables for a put to the
screen.
PRINTJOB
DO RED_Schd
DO WHILE .NOT. EOF()
    IF _plength = mlineno
        mlineno = 0
        EJECT PAGE
        DO RED_Schd
    ENDIF
    * Full page is advanced, line # count to zero, screen
    headers called.
    DO PHYS_DUE
    IF snext_PE <= report_date
        act_phys = "DUE"
    ELSE
        act_phys = ""
    ENDIF
    * Determines if physical is due during the date of this
    report.
    IF UPPER(blood) = "DUE"
        act_blood = "DUE"
    ELSE
        act_blood = ""
    ENDIF
    * Determines if blood type is unknown.

    IF UPPER(G6PD) = "DUE"
        act_G6PD = "DUE"
    ELSE
        act_G6PD = ""
    ENDIF
    * Determines if G6PD is due during the date of this
    report.

    IF UPPER(sicklecell) = "D"
        act_SC = "DUE"
    ELSE

```

```

        act_SC = ""
ENDIF
* Determines if sickle cell is due during the date of
this report.

DO PPD_DUE
IF next_PPD <= report_date .AND. next_PPD > {01/01/1912}
* Lower limit of 01/01/1912 is due to the fact that
11/11/11 is a
* temporary code date for allergic reaction and 12/12/12
is a temporary code date for immunization is due when
data was entered.
    act_PPD = "DUE"
ELSE
    act_PPD = ""
ENDIF
* Determines if PPD is due during the date of this
report.

DO YF_DUE
IF next_YF <= report_date .AND. next_YF > {01/01/1912}
    act_YF = "DUE"
ELSE
    act_YF = ""
ENDIF
* Determines if yellow fever is due during the date of
this report.

DO TD_DUE
IF next_TD <= report_date .AND. next_TD > {01/01/1912}
    act_TD = "DUE"
ELSE
    act_TD = ""
ENDIF
* Determines if tetanus is due during the date of this
report.

DO TYP_DUE
IF next_TYP <= report_date .AND. next_TYP > {01/01/1912}
.OR. BSC = .F.
    act_TYP = "DUE"
ELSE
    act_TYP = ""
ENDIF
* Determines if typhoid is due during the date of this
report.

IF act_phys = "DUE" .OR. act_blood = "DUE"
.OR. act_G6PD = "DUE";
.OR. act_SC = "DUE" .OR. act_PPD = "DUE"
.OR. act_YF = "DUE";

```

```

        .OR. act_TD = "DUE" .OR. act_TYP = "DUE"
        * When mcount is 1 a new command header is printed.
        DO TITLE
        DO RED_DATA
    ENDIF
    * Prints readiness information for everyone who
    * has at least one field requiring medical action.
    SKIP
ENDDO
* Loop until EOF marker is retrieved.
ENDPRINTJOB
DO PRN_RTRN
* Procedure to return print variables to their original
values.
ENDIF
SET ESCAPE ON
SET CENTURY OFF

```

```

*PROGRAM NAME:  READINS3
*PURPOSE      :  This is a report which displays:
*              1.  Name
*              2.  Rank(title)
*              3.  SSN
*              4.  Medical Action Required
*              of personnel who are due for at least one of the
*              following immunizations, physical exam and blood type.
*              --The program will query the user for the date of the
*              report and all medical action due prior to that date
*              will be printed.
*              --For individual curriculums.
*              --Individuals are grouped by their command.
*WRITTEN BY   :  Kevin Albert Bianchi
*LAST CHANGED:  30 JUL 93

```

```

SET TALK OFF
SET ESCAPE OFF
SET CENTURY ON
* Allows for 21ST century values in dates.
USE New_Data ORDER command
* use the new_data DBF with the command multiple index

GO TOP
* Set record pointer at the beginning of the DBF.
SET PRINTER ON

```

```

PUBLIC report_date, wrong_input, next_PE, next_PPD, next_YF,
next_TD,; next_TYP, ptitle, mprint, mcommand, snext_PE,
title, act_phys,; act_blood, act_G6PD, act_sikcel, act_ppd,

```

```

    act_yf, act_td, act_typ,; mlineno, good_report, report_day,
    report_month, report_year,; mpageno, mplength, curric_code,
    curric_name, selection
* Declares variables global.

* Variable Initialized
_pageno = 1
curric_code = SPACE(2)
report_month = SPACE(2)
report_day = 0
report_year = 0
wrong_input = 0
report_date = {}
good_report = "N"
next_PE = {}
next_PPD = {}
next_YF = {}
next_TD = {}
next_TYP = {}
mlineno = 0
mplength = 60
mpageno = 1
selection = SPACE(1)
* Variables initialized

DO WHILE UPPER(good_report) <> "Y"
    @ 5,10 SAY "This report will list the personnel who are due
    for medical"
    @ 6,12 SAY "action prior to the date you enter here on this
    screen."
    DO WHILE wrong_input < 1 .OR. wrong_input > 31
        @ 10,20 SAY "Enter the day of the month of this report."
        @ 12,6 SAY "****Enter the leading zero if value is 1-9
        (I.E. For 1 Jan ENTER 01). "
        @ 15,35 get report_day PICTURE "@Z 99"
        READ
        * Query the user for the cut off day of this report.

        wrong_input = report_day
        IF wrong_input < 1 .OR. wrong_input > 31
            @ 17,5 SAY "Incorrect input, the number entered must
            be a value between 1 and 31."
            @ 18,25 SAY "Press ENTER and try again."
            READ
        ENDIF
        * message put to the screen if the value entered for day
        is invalid
        CLEAR
    ENDDO
    * Loops until proper day value is input.

```

```

@ 10,4 SAY "Select the number equal to the month of your
report and press return."
@ 18,35 GET report_month PICTURE "@M
01,02,03,04,05,06,07,08,09,10,11,12";
MESSAGE "Press SPACEBAR to scroll, ENTER to select."
READ
CLEAR
* Query the user for the first month of the report
(multiple choice).

wrong_input = 0

DO WHILE wrong_input < YEAR(DATE())
@ 5,10 SAY "Enter the year which contains the month of
this report."
@ 7,14 SAY "***Enter all four numbers (I.E. For 1993
ENTER 1993)."
@ 12,35 GET report_year PICTURE "@Z 9999"
READ
* Query the user for the year of the first month of the
report.

wrong_input = report_year
IF wrong_input < YEAR(DATE())
@ 15,18 SAY "Incorrect input, the number entered must
be"
@ 16,18 SAY "a value greater than the current year's
date."
@ 18,25 SAY "Press ENTER and try again."
READ
ENDIF
CLEAR
* Message put to screen if the value for the year is too
small.
ENDDO
* Loops until proper year data is input.

@ 5,11 SAY "Select the curriculum or department code for
this report."
@ 7,19 SAY "The final option, 04, is for NPGS staff."
@ 10,35 GET curric_code PICTURE "@M
30,31,32,33,3A,34,35,36,37,38,39,04";
MESSAGE "Press SPACEBAR to scroll, ENTER to select."
READ
CLEAR
* Query user for curriculum or department code. Can be
expanded for other
* commands if division information is maintained for data.

report_date = CTOD(RIGHT(STR(report_day),2) + "/" +
report_month + "/" + RIGHT(STR(report_year),2))

```

```

DO CODE_NAM
ptitle = "MEDICAL READINESS REPORT FOR " +
        UPPER(curric_name) + " "; + DTOC(report_date)
* curric_name is the name of the curriculum returned from
CODE_NAME
* report_date is the date of the report

@ 5,18 SAY "The date of the report you selected is:"
@ 6,32 SAY DTOC(report_date)
@ 8,10 SAY "The curriculum you've chosen is " + curric_name
@ 12,18 SAY "Are these report specifications correct?"
@ 14,20 SAY "Select 'Y' for yes or 'N' for no."
@ 16,35 GET good_report PICTURE "@M Y,N";
        MESSAGE "Press SPACEBAR to scroll, ENTER to select"
READ
CLEAR
* Final query to ensure the proper date was input prior to
printing report.
ENDDO

@ 5,10 SAY "Do you want this report printed or put to the
screen?"
@ 7,17 SAY "Select 'P' for print or 'S' for screen."
@ 10,35 GET selection PICTURE "@M P,S";
        MESSAGE "Press SPACEBAR to scroll, ENTER to select"
READ
CLEAR
* Assigns a value to selection which determines a print or a
put to screen.

IF selection = "P"
PRINTJOB
DO RED_PgHd
DO WHILE .NOT. EOF()
    DO WHILE curric_div = curric_code
        IF mlineno = mplength
            mlineno = 0
            EJECT PAGE
            DO RED_PgHd
            DO RED_Cmnd
            ENDIF
            * Full page is advanced, line # count to zero and
            header programs called.

            DO PHYS_DUE
            IF snext_PE <= report_date
                act_phys = "DUE"
            ELSE
                act_phys = ""
            ENDIF
        ENDIF
    ENDIF
ENDIF

```

* Determines if physical is due during the date of this report.

```
IF UPPER(blood) = "DUE"
  act_blood = "DUE"
ELSE
  act_blood = ""
ENDIF
```

* Determines if blood type is unknown.

```
IF UPPER(G6PD) = "DUE"
  act_G6PD = "DUE"
ELSE
  act_G6PD = ""
ENDIF
```

* Determines if G6PD is due during the date of this report.

```
IF UPPER(sicklecell) = "D"
  act_SC = "DUE"
ELSE
  act_SC = ""
ENDIF
```

* Determines if sickle cell is due during the date of this report.

```
DO PPD_DUE
IF next_PPD <= report_date
  .AND. next_PPD > {01/01/1912}
  * The fact that 11/11/11 is a temporary code date for
  allergic reaction and 12/12/12 is a temporary code
  date for immunization
  * is taken care of in the called program, same for
  other shots.
```

```
  act_PPD = "DUE"
ELSE
  act_PPD = ""
ENDIF
```

* Determines if PPD is due during the date of this report.

```
DO YF_DUE
IF next_YF <= report_date
  .AND. next_YF > {01/01/1912}
  act_YF = "DUE"
ELSE
  act_YF = ""
ENDIF
```

* Determines if yellow fever is due during the date of this report.


```

DO TD_DUE
IF next_TD <= report_date
.AND. next_TD > {01/01/1912}
    act_TD = "DUE"
ELSE
    act_TD = ""
ENDIF
* Determines if tetanus is due during the date of
this report.

DO TYP_DUE
IF next_TYP <= report_date
.AND. next_TYP > {01/01/1912};
.OR. BSC = .F.
    act_TYP = "DUE"
ELSE
    act_TYP = ""
ENDIF
* Determines if typhoid is due during the date of
this report.

IF act_phys = "DUE" .OR. act_blood = "DUE"
.OR. act_G6PD = "DUE";
.OR. act_SC = "DUE" .OR. act_PPD = "DUE"
.OR. act_YF = "DUE";
.OR. act_TD = "DUE" .OR. act_TYP = "DUE"

    DO TITLE
    DO RED_DATA
ENDIF
* Prints readiness information for everyone who
* has at least one field requiring medical action.
SKIP
ENDDO
* Loop while curric_div does not change.
SKIP
ENDDO
* Loop until EOF marker is retrieved.
EJECT PAGE
ENDPRINTJOB
SET CENTURY OFF
ELSE
DO PRN_SCRN
* Procedure to prepare print variables for a put to screen.
PRINTJOB
DO RED_Schd
DO WHILE .NOT. EOF()
DO WHILE curric_div = curric_code
    IF _plength = mlineno
        mlineno = 0
        EJECT PAGE
    
```

```

DO RED_Schd
ENDIF
* Full page is advanced, line # count to zero and
header programs called.

DO PHYS_DUE
IF snext_PE <= report_date
    act_phys = "DUE"
ELSE
    act_phys = ""
ENDIF
* Determines if physical is due during the date of
this report.

IF UPPER(blood) = "DUE"
    act_blood = "DUE"
ELSE
    act_blood = ""
ENDIF
* Determines if blood type is unknown.

IF UPPER(G6PD) = "DUE"
    act_G6PD = "DUE"
ELSE
    act_G6PD = ""
ENDIF
* Determines if G6PD is due during the date of this
report.

IF UPPER(sicklecell) = "D"
    act_SC = "DUE"
ELSE
    act_SC = ""
ENDIF
* Determines if sickle cell is due during the date of
this report.

DO PPD_DUE
IF next_PPD <= report_date
    .AND. next_PPD > {01/01/1912}

    * The fact that 11/11/11 is a temporary code date for
    allergic reaction and 12/12/12 is a temporary code
    date for immunization is taken care of in the called
    program, same for other shots.
    act_PPD = "DUE"
ELSE
    act_PPD = ""
ENDIF
* Determines if PPD is due during the date of this
report.

```

```

DO YF_DUE
IF next_YF <= report_date
.AND. next_YF > {01/01/1912}
    act_YF = "DUE"
ELSE
    act_YF = ""
ENDIF
* Determines if yellow fever is due during the date
of this report.

DO TD_DUE
IF next_TD <= report_date .AND. next_TD > {01/01/12}
    act_TD = "DUE"
ELSE
    act_TD = ""
ENDIF
* Determines if tetanus is due during the date of
this report.

DO TYP_DUE
IF next_TYP <= report_date
.AND. next_TYP > {01/01/12} .OR. BSC = .F.
    act_TYP = "DUE"
ELSE
    act_TYP = ""
ENDIF
* Determines if typhoid is due during the date of
this report.

IF act_phys = "DUE" .OR. act_blood = "DUE"
.OR. act_G6PD = "DUE";
.OR. act_SC = "DUE" .OR. act_PPD = "DUE"
.OR. act_YF = "DUE";
.OR. act_TD = "DUE" .OR. act_TYP = "DUE"

    DO TITLE
    DO RED_DATA
ENDIF
* Prints readiness information for everyone who
* has at least one field requiring medical action.
SKIP
ENDDO
* Loop while curric_div does not change.
SKIP
ENDDO
* Loop until EOF marker is retrieved.
ENDPRINTJOB
SET CENTURY OFF
DO PRN_RTRN
* Procedure to return print variables to their original
value.

```

```
ENDIF
SET ESCAPE ON
SET CENTURY OFF
```

```
* PROGRAM NAME: Backup
* PURPOSE      : Back up NEW_DATA.DBF to NEWDATBK.DBF for
                  backup protection.
* WRITTEN BY   : KEVIN ALBERT BIANCHI
* LAST CHANGED: 20 Aug 93
```

```
SET ESCAPE OFF
```

```
USE new_data.dbf ORDER ssno
@ 5,15 SAY "Select OVERWRITE and wait for the menu to return."
@ 13,37 SAY "Hi!"
COPY TO newdatbk.dbf WITH PRODUCTION
* Select overwrite statement instructs the user to select
  overwrite when
* dBASE IV asks for overwrite or cancel.
SET ESCAPE ON
```

```
* PROGRAM NAME: Code_Nam
* PURPOSE      : Given the curriculum code this will return the
                  curriculum name.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 30 JUL 93
```

```
PROCEDURE CODE_NAM
*   PARAMETERS curric_code
*   PRIVATE
```

```
DO CASE
CASE curric_code = "30"
    curric_name = "Operations Analysis"
CASE curric_code = "31"
    curric_name = "Aerospace Engineering"
CASE curric_code = "32"
    curric_name = "Electrical & Computer Engineering"
CASE curric_code = "33"
    curric_name = "Weapons Engineering"
CASE curric_code = "3A" .OR. curric_code = "3a"
    curric_name = "ASW & Electronic Warfare"
CASE curric_code = "34"
    curric_name = "Naval Engineering"
CASE curric_code = "35"
    curric_name = "Air-Ocean Science"
CASE curric_code = "36"
    curric_name = "Administrative Science"
```

```

CASE curric_code = "37"
    curric_name = "Computer Technology"
CASE curric_code = "38"
    curric_name = "National Security Affairs"
CASE curric_code = "39"
    curric_name = "Joint C3 and Space"
CASE curric_code = "04"
    curric_name = "NPGS Staff Personnel"
ENDCASE
* All cases assign the curriculum name from a given curriculum
code.
RETURN

```

```

* PROGRAM NAME: DUE_DATA
* PURPOSE      : Compiles the output data for the PFT report
*               when 12/12/12 is entered for the date of the
*               physical exam.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 13 AUG 93

```

PROCEDURE Due_Data

```

? lastname PICTURE "@T XXXXXXXXXXXXXXXXXXXX" AT 0, " , "
?? firstname PICTURF "@T XXXXXXXXXXXX", " ", initial
PICTURE "X",
?? title PICTURE "@T XXXXXX" AT 35, SSNO PICTURE
"XXXXXXXXXX" AT 43,
?? physical PICTURE "@E 99/99/99" AT 56, "DUE" AT 68
mlineno = mlineno + 1
* PFT report data line printed and line # counter advanced
accordingly.

```

RETURN

```

* PROGRAM NAME : Dupechk
* PURPOSE      : Prevents duplicate data entries by doing a
                SEEK on the SSN field.
* WRITTEN BY   : Boreland
* LAST CHANGED : 17 JUL 93 (adapted for this application)

```

FUNCTION Dupechk

```

PARAMETER lc_dupefield, lc_tag
* Field and index.
ln_select = SELECT()
* Identifies the work area.
USE DBF() ORDER (lc_tag) AGAIN IN (ln_select)

```

```

* Opens DBF and uses current DBF indexed mem-var assigned
* to lc_tag.
ll_dupe = SEEK(lc_dupefield, ln_select)
* ll_dupe assigned to duplicate field.
USE IN (ln_select)
* Closes DBF in the work area.
RETURN .NOT. ll_dupe
* Returns non_duplicate field value.

```

```

* PROGRAM NAME:  HIV_Cmnd
* PURPOSE       :  Header for the HIV reports.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  09 JUL 1993

```

```

PROCEDURE HIV_Cmnd

```

```

    IF mlineno + 5 > _plength
        EJECT PAGE
        DO HIV_PgHd
        ?
        ? "***Command: " AT 0, command Style "B" AT 13
        mlineno = mlineno + 2
        * If there's only enough room for command header and two
        records
        * the page is ejected and the page header program is
        called
        * followed by a command header and line # counter.

    ELSE
        ?
        ? "***Command: " AT 0, command STYLE "B" AT 13
        mlineno = mlineno + 2
        * Command header printed and the line # counter advanced
        accordingly.

    ENDIF
RETURN

```

```

* PROGRAM NAME:  HIV_Data
* PURPOSE       :  Compiles the output data for the Checkout
                  reports (OUT_NOW and OUT_NEXT).
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  08 AUG 93

```

```

PROCEDURE HIV_Data

```

```

    ? lastname PICTURE "@T XXXXXXXXXXXXXXXXXXXXX" AT 0, ", "

```

```

?? firstname PICTURE "@T XXXXXXXXXXXXX", " ", initial
PICTURE "X",
?? title PICTURE "@T XXXXXX" AT 33, SSNO PICTURE
"XXXXXXXXXX" AT 41,
?? command PICTURE "@T XXXX" AT 53, curric_div PICTURE "XX"
AT 61
?? HIV PICTURE "@E 99/99/99" AT 67
mlineno = mlineno + 1
* Out_data report info printed and line # counter advanced.

```

RETURN

```

* PROGRAM NAME: HIV_PgHd
* PURPOSE      : Page header for the OUT_NOW and OUT_NEXT
                  reports.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 08 AUG 1993

```

PROCEDURE HIV_PgHd

```

_wrap = .T.
* wrap must be on for "center" to work.

_alignment = "center"
? ptitle
?
?
?
_alignment = "LEFT"
? LTRIM(STR(DAY(DATE())) AT 0
?? CMONTH(DATE()) AT 3
?? LTRIM(STR(YEAR(DATE())) AT (4 + LEN(CMONTH(DATE()))))
?? "Page " AT 69, LTRIM(STR(mpageno,2,0)) AT 74
?
? "Name" STYLE "U" AT 1, "Rank" STYLE "U" AT 33, "SSN"
STYLE "U" AT 41
?? "CMND" STYLE "U" AT 53, "Cur/Div" STYLE "U" AT 59,
"HIV" STYLE "U" AT 69
?
mlineno = mlineno + 8
mpageno = mpageno + 1
* Page title, date, page #, and column headers printed.
RETURN

```

```

* PROGRAM NAME:  HIV_ScHd
* PURPOSE       :  Page header for the checkout reports to the
                   screen.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  20 AUG 1993

```

PROCEDURE HIV_ScHd

```

_wrap = .T.
* wrap must be on for "center" to work.

  _alignment = "center"
  ? ptitle
  * Determined by the calling program.
  ?
  _alignment = "LEFT"
  ? "Name" STYLE "U" AT 1, "Rank" STYLE "U" AT 33, "SSN"
    STYLE "U" AT 41
  ?? "CMND" STYLE "U" AT 53, "Cur/Div" STYLE "U" AT 59,
    "HIV" STYLE "U" AT 69
  mlineno = mlineno + 3
  mpageno = mpageno + 1
  * Page title, a blank line, and column headers printed.
RETURN

```

```

* PROGRAM NAME:  Locate
* PURPOSE       :  Retrieves a record after querying the user
                   for the SSNO.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  19 JUL 93

```

```

SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER SSNO
* Use the new_data DBF with an index on ssno.

```

```

GO TOP
* Record pointer to the beginning of the DBF.

```

```

PRIVATE MSSNO
MSSNO = SPACE (11)
* declare MSSNO as a local, character string variable of 11
characters.

```

```

@ 5,15 SAY "Enter the Social Security Number of the"
@ 6,15 SAY "individual whose record you want to edit."
@ 8,13 SAY "Nine numbers separated by conventional dashes"
@ 9,23 SAY "(i.e. 123-45-6789)"
@ 11,25 GET MSSNO PICTURE "999-99-9999"

```



```

READ
CLEAR
@ 10,3 SAY "After editing a record Press CONTROL and END
together to save changes made"
@ 11,8 SAY "otherwise press ESCAPE both actions will return
you to the menu."
@ 15,26 SAY "Press ENTER now to continue."
* Query user for a SSNO and assigns it to MSSNO.

```

```

READ
CLEAR
SEEK MSSNO
* Record is searched for MSSNO and put into an edit screen.
* If the record does not exist the message below is put to the
screen.
IF FOUND() =.T.
    EDIT
ELSE
    @ 5,20 SAY "This record is not in the database."
    @ 15,26 SAY "Press ENTER now to continue."
    READ
    CLEAR
ENDIF

```

```

@ 5,4 SAY "If you did not press CONTROL and END together all
changes were not saved."
@ 15,26 SAY "Press ENTER now to continue."
READ
CLEAR
SET ESCAPE ON

```

```

* PROGRAM NAME:  OUT_Cmnd
* PURPOSE       :  Header for each Command in the Checkout
                  reports
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  09 JUL 1993

```

```

PROCEDURE OUT_Cmnd

```

```

    IF mlineno + 5 > _plength
        EJECT PAGE
        DO OUT_PgHd
        ?
        ? "****Command: " AT 0, command Style "B" AT 13
        mlineno = mlineno + 2
        * If there's only enough room for command header and two
        records
        * the page is ejected and the page header program is
        called

```

```

    * followed by a command header and line # counter.
ELSE
    ?
    ? "****Command: " AT 0, command STYLE "B" AT 13
    mlineno = mlineno + 2
    * Command header printed and the line # counter advanced
    accordingly.

ENDIF
RETURN

```

```

* PROGRAM NAME:  OUT_Data
* PURPOSE       :  Compiles the output data for the Checkout
                   reports.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  09 JUN 93

```

PROCEDURE OUT_Data

```

    ? lastname PICTURE "@T XXXXXXXXXXXXXXXXXXXX" AT 0, ", "
    ?? firstname PICTURE "@T XXXXXXXXXXXX", " ", initial
    PICTURE "X",
    ?? title PICTURE "@T XXXXXX" AT 33, SSNO PICTURE
    "XXXXXXXXXX" AT 41,
    ?? command PICTURE "@T XXXX" AT 57, checkout PICTURE "@E
    99/99/99" AT 64
    mlineno = mlineno + 1
    * Out_data report info printed and line # counter advanced.

RETURN

```

```

* PROGRAM NAME:  Out_Mark
* PURPOSE       :  Marks files for deletion.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  19 JUL 93

```

```

SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER SSNO
* Use the new_data DBF with an index on ssno.

```

```

* variables initialized
PRIVATE MSSNO, record_no, mcontinue
MSSNO = SPACE (11)
* declare MSSNO as a local, character string variable of 11
characters.
record_no = 0

```

```

mcontinue = "Y"
*variables initialized

DO WHILE mcontinue = "Y"
  GO TOP
  * Record pointer to the beginning of the DBF.

  @ 5,20 SAY "Enter the Social Security Number of the"
  @ 6,21 SAY "record you want to mark for deletion."
  @ 8,17 SAY "Nine numbers separated by conventional dashes"
  @ 9,31 SAY "(i.e. 123-45-6789)"
  @ 11,34 GET MSSNO PICTURE "999-99-9999"
  * Query user for a SSNO and assigns it to MSSNO.

  READ
  CLEAR
  FIND &MSSNO
  * Record is located
  READ
  IF FOUND() = .T.
    record_no = RECNO()
    mname = lastname
    DELETE RECORD record_no
    @ 5,1 SAY mname + "marked for deletion (press ENTER to
    continue). "
    READ
    CLEAR
    * Record # is retrieved and the record is marked for
    deletion.
  ELSE
    @ 5,20 SAY "This record is not in the database."
    READ
    CLEAR
  ENDIF

  @ 5,15 SAY "Do you want to mark another record for
  deletion?"
  @ 7,20 SAY "Select 'Y' for yes and 'N' for no."
  @ 10,35 GET mcontinue PICTURE "@M Y,N";
  MESSAGE "Press SPACEBAR to scroll, ENTER to select"
  * Query user to continue marking files else exit program.

  READ
  CLEAR
ENDDO
SET ESCAPE ON

```

```

* PROGRAM NAME:  OUT_PgHd
* PURPOSE       :  Page header for the checkout reports.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  09 JUN 1993

```

PROCEDURE OUT_PgHd

```

_wrap = .T.
* wrap must be on for "center" to work.

  _alignment = "center"
  ? ptitle
  ?
  ?
  ?
  _alignment = "LEFT"
  ? LTRIM(STR(DAY(DATE())) AT 0
  ?? CMONTH(DATE()) AT 3
  ?? LTRIM(STR(YEAR(DATE())) AT (4 + LEN(CMONTH(DATE()))
  ?? "Page " AT 69, LTRIM(STR(mpageno,2,0)) AT 74
  ?
  ? "Name" STYLE "U" AT 1, "Rank" STYLE "U" AT 33, "SSN"
  STYLE "U" AT 41
  ?? "CMND" STYLE "U" AT 57, "Checkout" STYLE "U" AT 64
  ?
  mlineno = mlineno + 8
  mpageno = mpageno + 1
  * Page title, date, page #, and column headers printed.
RETURN

```

```

* PROGRAM NAME:  OUT_ScHd
* PURPOSE       :  Page header for the checkout reports.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  09 JUN 1993

```

PROCEDURE OUT_ScHd

```

_wrap = .T.
* wrap must be on for "center" to work.

  _alignment = "center"
  ? ptitle
  * Determined by the calling program.
  ?
  _alignment = "LEFT"
  ? "Name" STYLE "U" AT 1, "Rank" STYLE "U" AT 33, "SSN"
  STYLE "U" AT 41
  ?? "CMND" STYLE "U" AT 57, "Checkout" STYLE "U" AT 64
  mlineno = mlineno + 3

```

```
    mpageno = mpageno + 1
    * Page title, a blank line, and column headers printed.
RETURN
```

```
* PROGRAM NAME: Out_UMrk
* PURPOSE      : Un-marks files which were marked for deletion.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 29 JUL 93
```

```
SET TALK OFF
SET ESCAPE OFF
USE New_Data ORDER SSNO
* Use the new_data DBF with an index on ssno.
```

```
*variables
PRIVATE MSSNO, record_no, mcontinue
MSSNO = SPACE (11)
* declare MSSNO as a local, character string variable of 11
characters.
record_no = 0
mcontinue = "Y"
*variables
```

```
DO WHILE mcontinue = "Y"
    GO TOP
    * Record pointer to the beginning of the DBF.

    @ 5,20 SAY "Enter the Social Security Number of the"
    @ 6,21 SAY "record you want to un-mark for deletion."
    @ 8,17 SAY "Nine numbers separated by conventional dashes"
    @ 9,31 SAY "(i.e. 123-45-6789)"
    @ 11,34 GET MSSNO PICTURE "999-99-9999"
    * Query user for a SSNO and assigns it to MSSNO.
```

```
    READ
    CLEAR
    FIND &MSSNO
    * Record is located
```

```
    READ
    IF FOUND() = .T.
        record_no = RECNO()
        mname = lastname
        RECALL RECORD record_no
        @ 5,1 SAY mname + "was unmarked for deletion (press
        ENTER to continue)."
        READ
        CLEAR
        * Record # is retrieved and the record is marked for
        deletion.
```

```

ELSE
  ● 5,20 SAY "This record is not in the database."
  READ
  CLEAR
ENDIF

● 5,13 SAY "Do you want to un-mark another record for
deletion?"
● 7,20 SAY "Select 'Y' for yes and 'N' for no."
● 10,35 GET mcontinue PICTURE "@M Y,N";
  MESSAGE "Press SPACEBAR to scroll, ENTER to select"
  * Query user to continue marking files else exit program.

READ
CLEAR
ENDDO
SET ESCAPE ON

```

```

* PROGRAM NAME: PE_Due_1
* PURPOSE      : Returns a logical true for the memory variable
                  due if the physical exam due date is due by
                  current date.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 6 JUL 93

```

```
PROCEDURE PE_DUE_1
```

```

IF MONTH(snext_PE) <= MONTH(DATE())
  .AND. YEAR(snext_PE) = YEAR(DATE());
  .OR. YEAR(snext_PE) < YEAR(DATE())
    due = .T.
ELSE
  due = .F.
ENDIF
* If the individuals physical exam was due before the current
date a logical
* true is assigned to due.

RETURN

```

```

* PROGRAM NAME: PFT_Cmdnd
* PURPOSE      : Header for each Command in the PFT reports.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 07 JUL 1993

```

```
PROCEDURE PFT_Cmdnd
```

```

IF mlineno + 5 > _plength
  EJECT PAGE
  mlineno = 0
  DO PFT_PgHd
  ?
  ? "****Command: " AT 0, command Style "B" AT 13
  mlineno = mlineno + 2
  * Advances page, line # counter to zero, page header
  program called
  * command line printed, and line # counter advanced
  accordingly.
ELSE
  ?
  ? "****Command: " AT 0, command STYLE "B" AT 13
  mlineno = mlineno + 2
  * Command printed and line # counter advanced
ENDIF
RETURN

```

```

* PROGRAM NAME:   PFT_PgHd
* PURPOSE       :   Page header for the PFT reports.
* WRITTEN BY    :   Kevin Albert Bianchi
* LAST CHANGED:  22 June 1993

```

```

PROCEDURE PFT_PgHd

```

```

  _wrap = .T.
  * Wrap must be on for center to work.

  _alignment = "center"
  ? ptitle
  ?
  ?
  ?
  _alignment = "LEFT"
  ? LTRIM(STR(DAY(DATE())) AT 0
  ?? CMONTH(DATE()) AT 3
  ?? LTRIM(STR(YEAR(DATE())) AT (4 + LEN(CMONTH(DATE()))
  ?? "Page " AT 69, LTRIM(STR(mpageno,2,0)) AT 74
  ?
  ? "Name" STYLE "U" AT 1, "Rank" STYLE "U" AT 35, "SSN"
  STYLE "U" AT 43
  ?? "Last_PE" STYLE "U" AT 56, "Next_PE" STYLE "U" AT 66
  ?
  mlineno = mlineno + 8
  mpageno = mpageno + 1
  * Title, date, page #, data printed, and the line # counter
  advanced.
RETURN

```

```

* PROGRAM NAME:  PFT_ScHd
* PURPOSE       :  Page header for the PFT reports put to the
                   screen.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  20 Aug 93

```

PROCEDURE PFT_ScHd

```

_wrap = .T.
* Wrap must be on for center to work.

  _alignment = "center"
  ? ptitle
  * Determined by the calling program.
  ?
  _alignment = "LEFT"
  ? "Name" STYLE "U" AT 1, "Rank" STYLE "U" AT 35, "SSN"
  STYLE "U" AT 43
  ?? "Last_PE" STYLE "U" AT 56, "Next_PE" STYLE "U" AT 66
  mlineno = mlineno + 3
  mpageno = mpageno + 1
  * Title, header data printed, and the line # counter
  advanced.

```

RETURN

```

*PROGRAM NAME:  Phys_Due
*PURPOSE       :  This will determine the date of which an
                   individual's current physical will expire.
*WRITTEN BY    :  Kevin Albert Bianchi
*LAST CHANGED:  19 June 1993

```

PROCEDURE PHYS_DUE

```

*Variables:  flight, physical, DOB, next_PE
next_PE = {}
* Next physical exam due mem_var.

IF flight = .T.
  * Flight designated.
  IF physical < DATE() - 62
    IF MONTH(DATE()) <= MONTH(DOB)
      next_PE = CTOD(STR(DAY(DOB)) + "/" +
                  STR(MONTH(DOB)) + "/" +
                  +RIGHT(STR(YEAR(DATE())),2))
      * Next_PE is this years birthday.
    ELSE
      next_PE = CTOD(STR(DAY(DOB)) + "/" +
                  STR(MONTH(DOB)) + "/");

```



```

        + RIGHT(STR(YEAR( DATE() ) + 1),2))
    * Next_PE is next years birthday.
ENDIF
    * Physical not in the last 2 months.
ELSE
    * Physical done within the last two months.
    IF MONTH(physical) = 12 .AND. MONTH(DOB) = 1
        .AND. MONTH( DATE() ) = 12
        next_PE = CTOD(STR(DAY(DOB)) + "/" +
            STR(MONTH(DOB)) + "/" ;
            + RIGHT(STR(YEAR( DATE() ) + 2),2))
        * Next_PE when last physical was in December and
        DOB is January.
    ELSE
        next_PE = CTOD(STR(DAY(DOB)) + "/" +
            STR(MONTH(DOB)) + "/" ;
            + RIGHT(STR(YEAR( DATE() ) + 1),2))
        * Next_PE is next years birthday.
    ENDIF
ENDIF
ELSE
    * Not flight designated
    IF YEAR( DATE() ) - 25 < YEAR(DOB) .OR. ;
        YEAR( DATE() ) - 25 = YEAR(DOB) .AND.
        MONTH( DATE() ) < MONTH(DOB)
    * Determines if the individual is less than 25
    IF YEAR( DATE() ) - 5 < YEAR(physical) .OR. ;
        YEAR( DATE() ) - 5 = YEAR(physical) .AND.
        MONTH( DATE() ) < MONTH(DOB)
    * Determines if the current physical was done
    * prior to the individuals 25th birthday
        next_PE = CTOD(STR(DAY(physical)) + "/" +
            STR(MONTH(physical)) ;
            + "/" + RIGHT(STR(YEAR(physical) + 5),2))
        * Next_PE is 5 years from last physical.
    ELSE
        next_PE = CTOD(STR(DAY(DOB)) + "/" +
            STR(MONTH(DOB)) + "/" ;
            + RIGHT(STR(YEAR(DOB) + 25),2))
        * Next_PE is the individuals 25th birthday.
    ENDIF
ELSE
    next_PE = CTOD(STR(DAY(physical)) + "/" +
        STR(MONTH(physical)) ;
        + "/" + RIGHT(STR(YEAR(physical) + 5),2))
    * Next_PE is 5 years from last physical.
ENDIF
ENDIF
snext_PE = next_PE - DAY(next_PE) + 1
* makes physical due date the first day of the month.
RETURN

```

```

*PROGRAM NAME: PPD_Due
*PURPOSE      : This will determine the date which
*              individuals' are due for their PPD shot.
*WRITTEN BY   : Kevin Albert Bianchi
*LAST CHANGED: 13 JUL 1993

```

PROCEDURE PPD_DUE

```

IF PPD = {11/11/11}
  next_PPD = DATE() + 1095
ELSE
  next_PPD = CTOD("01/" + STR(MONTH(PPD)-1) + "/" +
    RIGHT(STR(YEAR(PPD)+3),2))
ENDIF
* 11/11/11 gets 3 years added because it is the symbol for do
not do (allergic)
* otherwise next_PPD is the first day of the month completed
+ 3 years
RETURN

```

```

* PROGRAM NAME: PRN_RTRN
* PURPOSE      : Returns output which was Prepared to be put to
                 the screen
*              back to printer settings.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 15 Aug 93

```

PROCEDURE prn_rtrn

```

RESTORE FROM scrnfile.mem ADDITIVE
  _plength = mpscrnlength
  _pwait = mpwait
  _peject = mpeject
SET PRINTER TO LPT1
ERASE scrnfile.mem
* Returns system variables and settings to their original
state.

RETURN

```

```

* PROGRAM NAME: PRN_SCRN
* PURPOSE      : Prepares output to be put to the screen.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 15 Aug 93

```

PROCEDURE prn_scrn

```

PUBLIC mplength, mpwait, mpeject

```

```

* Variables
mpscrnlength = 60
mpwait = .F.
mpeject = "BEFORE"

_plength = mpscrnlength
_pwait = mpwait
_peject = mpeject
SAVE ALL LIKE mp* TO scrnfile.mem

_plength = 20
_pwait = .T.
_peject = "none"
SET PRINTER TO NUL
* Prepares the report to go to the screen

RETURN

```

```

* PROGRAM NAME:  RED_Cmnd
* PURPOSE      :  Header for each Command for the readiness
                  reports.
* WRITTEN BY   :  Kevin Albert Bianchi
* LAST CHANGED:  12 JUL 1993

```

```

PROCEDURE RED_Cmnd

```

```

    IF mlineno + 5 > _plength
        EJECT PAGE
        DO RED_PgHd
        ?
        ? "****Command: " AT 0, command Style "B" AT 13
        mlineno = mlineno + 2
        * If less than 3 records will be under the new command
        header, page is
        * ejected, page header is called, command header and line
        # count is done.
    ELSE
        ?
        ? "****Command: " AT 0, command STYLE "B" AT 13
        mlineno = mlineno + 2
        * command header and line adjustment
    ENDIF

RETURN

```

```

* PROGRAM NAME:  RED_Data
* PURPOSE       :  Compiles the output data for readiness
                   reports.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  12 Jul 93

```

PROCEDURE RED_Data

```

? lastname PICTURE "@T XXXXXXXXXXX" AT 0, firstname PICTURE
"@T X"
?? " ", initial PICTURE "X", title PICTURE "@T XXXXXX" AT
18
?? SSNO PICTURE "XXXXXXXXXXXX" AT 25, act_phys PICTURE "AAA"
AT 37
?? act_blood PICTURE "AAA" AT 42, act_G6PD PICTURE "AAA" AT
48
?? act_SC PICTURE "AAA" AT 53, act_PPD PICTURE "AAA" AT 58
?? act_YF PICTURE "AAA" AT 63, act_TD PICTURE "AAA" AT 68
?? act_TYP PICTURE "AAA" AT 73
mlineno = mlineno + 1
* Readiness report data line and line # counter.
RETURN

```

```

* PROGRAM NAME:  RED_PgHd
* PURPOSE       :  Page header for the readiness reports.
* WRITTEN BY    :  Kevin Albert Bianchi
* LAST CHANGED:  12 JUL 1993

```

PROCEDURE RED_PgHd

```

_wrap = .T.
*wrap must be on for center alignment to work.

_alignment = "center"
? ptitle
* Determined by the calling program.
?
?
?
_alignment = "LEFT"
? LTRIM(STR(DAY(DATE())) AT 0
?? CMONTH(DATE()) AT 3
?? LTRIM(STR(YEAR(DATE()))) AT (4 + LEN(CMONTH(DATE())))
?? "Page " AT 69, LTRIM(STR(mpageno,2,0)) AT 74
?
? "Name" STYLE "U" AT 0, "Rank" STYLE "U" AT 18, "SSN"
STYLE "U" AT 25
?? "PHYS" STYLE "U" AT 37, "BLOOD" STYLE "U" AT 42, "G6PD"
STYLE "U" AT 48

```

```

?? "SC" STYLE "U" AT 53, "PPD" STYLE "U" AT 58, "YF" STYLE
"U" AT 63
?? "TD" STYLE "U" AT 68, "TYP" STYLE "U" AT 73
?
mlineno = mlineno + 8
mpageno = mpageno + 1
* Prints title, three blank lines, date, page no, and
column headers.
RETURN

```

```

* PROGRAM NAME: RED_Schd
* PURPOSE      : Page header for the READINES reports put to
                  the screen.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 20 AUG 1993

```

PROCEDURE RED_Schd

```

_wrap = .T.
*wrap must be on for center alignment to work.

_alignment = "center"
? ptitle
?
* Determined by the calling program.
_alignment = "LEFT"
? "Name" STYLE "U" AT 0, "Rank" STYLE "U" AT 18, "SSN"
STYLE "U" AT 25
?? "PHYS" STYLE "U" AT 37, "BLOOD" STYLE "U" AT 42, "G6PD"
STYLE "U" AT 48
?? "SC" STYLE "U" AT 53, "PPD" STYLE "U" AT 58, "YF" STYLE
"U" AT 63
?? "TD" STYLE "U" AT 68, "TYP" STYLE "U" AT 73
mlineno = mlineno + 3
mpageno = mpageno + 1
* PUTS title, a blank line, and column headers to the
screen.
RETURN

```

```

* PROGRAM NAME: Restore
* PURPOSE      : Restores NEWDATBK.DBF to NEW_DATA.DBF.
* WRITTEN BY   : Kevin Albert Bianchi
* LAST CHANGED: 20 AUG 93

```

SET ESCAPE OFF

USE newdatbk.dbf ORDER ssno

● 5,15 SAY "Select OVERWRITE and wait for the menu to return."
● 13,37 SAY "Hi!"
COPY TO new_data.dbf WITH PRODUCTION
USE new_data.dbf
* Copies backup file to the main application DBF.

SET ESCAPE ON

*PROGRAM NAME: TD_Due (Tetanus)
*PURPOSE : This will determine the date of which
* individuals' are due for their Tetanus shot.
*WRITTEN BY : Kevin Albert Bianchi
*LAST CHANGED: 13 JUL 1993

PROCEDURE TD_DUE

IF tetanus = {11/11/11}
next_TD = DATE() + 1095
ELSE
next_TD = CTOD("01/" + STR(MONTH(tetanus)-1) + "/" ;
+ RIGHT(STR(YEAR(tetanus)+10),2))
ENDIF
* 11/11/11 gets 3 years added because it is the symbol for do
not do (allergic)
* other wise next_TD is the first day of the month last given
+ 10 years.

RETURN

* PROGRAM NAME: TITLE (Rank/Service)
* PURPOSE : Given an individual's military RANK (I.E. 0-1)
* and SERVICE their service specific title is
* determined (I.E. Ensign).
* WRITTEN BY : Kevin Albert Bianchi
* LAST CHANGED: 6 JUL 93

PROCEDURE TITLE

*Rank is specified by military pay-grade (I.E. E-1).
*Service is either USN or USMC.

IF UPPER(service) = "USN"
DO CASE
CASE UPPER(rank) = "E-1"
title = "SR"
CASE UPPER(rank) = "E-2"
title = "SA"
CASE UPPER(rank) = "E-3"

```

        title = "SN"
CASE UPPER(rank) = "E-4"
    title = "PO3"
CASE UPPER(rank) = "E-5"
    title = "PO2"
CASE UPPER(rank) = "E-6"
    title = "PO1"
CASE UPPER(rank) = "E-7"
    title = "CPO"
CASE UPPER(rank) = "E-8"
    title = "SCPO"
CASE UPPER(rank) = "E-9"
    title = "MCPO"
CASE UPPER(rank) = "O-1"
    title = "ENS"
CASE UPPER(rank) = "O-2"
    title = "LTJG"
CASE UPPER(rank) = "O-3"
    title = "LT"
CASE UPPER(rank) = "O-4"
    title = "LCDR"
CASE UPPER(rank) = "O-5"
    title = "CDR"
CASE UPPER(rank) = "O-6"
    title = "CAPT"
CASE UPPER(rank) = "O-7"
    title = "RADM"
CASE UPPER(rank) = "O-8"
    title = "RADM"
CASE UPPER(rank) = "O-9"
    title = "VADM"
ENDCASE
ELSE
    IF UPPER(service) = "USMC"
        DO CASE
            CASE UPPER(rank) = "E-1"
                title = "PVT"
            CASE UPPER(rank) = "E-2"
                title = "PFC"
            CASE UPPER(rank) = "E-3"
                title = "LCPL"
            CASE UPPER(rank) = "E-4"
                title = "CPL"
            CASE UPPER(rank) = "E-5"
                title = "SGT"
            CASE UPPER(rank) = "E-6"
                title = "SSGT"
            CASE UPPER(rank) = "E-7"
                title = "GYSGT"
            CASE UPPER(rank) = "E-8"
                title = "MSGT"

```

```

CASE UPPER(rank) = "E-9"
    title = "MGYSGT"
CASE UPPER(rank) = "O-1"
    title = "2LT"
CASE UPPER(rank) = "O-2"
    title = "1LT"
CASE UPPER(rank) = "O-3"
    title = "CAPT"
CASE UPPER(rank) = "O-4"
    title = "MAJ"
CASE UPPER(rank) = "O-5"
    title = "LCOL"
CASE UPPER(rank) = "O-6"
    title = "COL"
CASE UPPER(rank) = "O-7"
    title = "BGEN"
CASE UPPER(rank) = "O-8"
    title = "MGEN"
CASE UPPER(rank) = "O-9"
    title = "LTGEN"
ENDCASE
* All cases assign naval title to it's perspective
generic rank.
ELSE
    title = " "
    * Tile is null if data is faulty.
ENDIF
ENDIF
RETURN

```

```

*PROGRAM NAME:  TYP_Due
*PURPOSE       :   This will determine the date of which
*               individuals' are due for their Typhoid shot.
*WRITTEN BY    :   Kevin Albert Bianchi
*LAST CHANGED:  13 JUL 1993

```

PROCEDURE TYP_DUE

```

IF typhoid = {11/11/11}
    next_TYP = DATE() + 1095
ELSE
    next_TYP = CTOD("01/" + STR(MONTH(typhoid)-1) + "/" +
    + RIGHT(STR(YEAR(typhoid)+3),2))
ENDIF
* 11/11/11 gets 3 years added because it is the symbol for do
not do (allergic)
* otherwise next_TYP is the first day of the month completed
+ 3 years.
RETURN

```

*PROGRAM NAME: YF_Due
*PURPOSE : This will determine the date of which
* individuals' are due for their Yellow fever
shot.
*WRITTEN BY : Kevin Albert Bianchi
*LAST CHANGED: 13 JUL 1993

PROCEDURE YF_DUE

IF yellowfev = {11/11/11}
next_YF = DATE() + 1095
ELSE
next_YF = CTOD("01/" + STR(MONTH(yellowfev)-1) + "/" ;
+ RIGHT(STR(YEAR(yellowfev)+10),2))
ENDIF
* 11/11/11 gets 3 years added because it is the symbol for do
no do (allergic) other wise next_YF is the 1st day of the
month + 10 years.

RETURN

APPENDIX E: USER'S MANUAL
TO THE NAVY MEDICAL ADMINISTRATIVE UNIT
DATABASE SYSTEM

I. INTRODUCTION

The purpose of this manual is to provide a step by step explanation of the Navy Medical Administrative Unit Database System (NMAUDS 1.0). This manual is provided solely for the purpose of using NMAUDS 1.0. Any maintenance of the system will require the thesis document provided and/or knowledge of dBASE IV version 1.5. For further guidance in dBASE IV refer to the manuals issued with dBASE IV software.

II. STARTING NMAUDS

There are three recommended methods to initiate NMAUDS.

- (1) From a DOS batch file that will access the dBASE subdirectory, start dBASE IV, and execute MEDICAL.APP.
- (2) From a menu system which will execute the batch file previously described in option (1).
- (3) From the dbase dot prompt by typing [DO MEDICAL.APP].

NMAUDS.BAT is the batch file originally dispensed with NMAUDS 1.0.

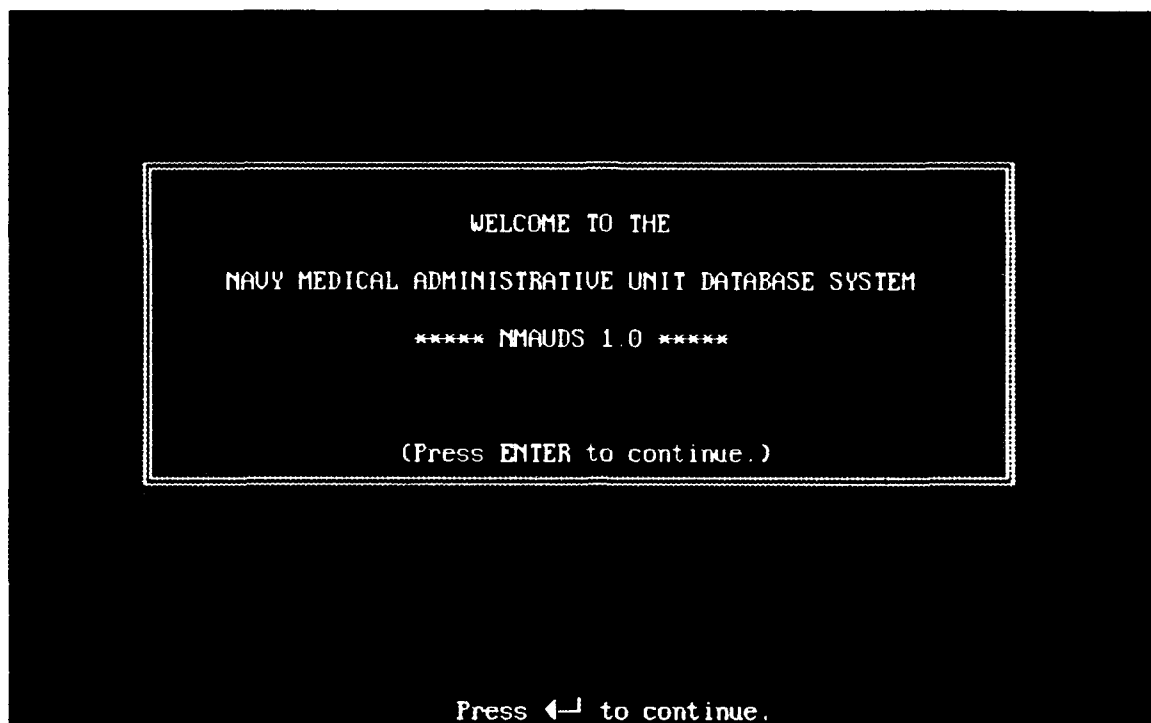


Figure F-1 Welcome Screen

The first screen displayed in NMAUDS is the welcome screen Figure F-1. To continue beyond the welcome screen press <ENTER>.

III. MENU OPERATION

The first menu in the system, the main menu, illustrated in Figure F-2 provides three options. The options can be highlighted with the arrow keys and selected by pressing <ENTER> while the desired option is highlighted. All three selections produce additional menus. Deselecting these menus is accomplished by pressing <ESC> or <Escape>.



Figure F-2 Main Menu

A. REPORTS MENU

Figure F-3 is an illustration of the reports menu. The options on this menu can be highlighted with the arrow keys and selected by pressing <ENTER> while the desired option is highlighted. The first three selections on this menu exhibit an additional menu and they are discussed in sub-sections 1-3 below. All reports query the operator for necessary report parameters. Report parameter inquiries are self explanatory.

The fourth option on this menu is **Morbidity Report**. When selected, by pressing <ENTER>, the morbidity program is executed. The program will retrieve from the user the month of the report.

Reports	Update Data	Exit
Physicals Needed for PFT Checkout Reports Medical Readiness Reports Morbidity Report Filing Report (Shelf-list) HIU Report NPGS personnel lacking a code -----		

Use the arrow keys to highlight, ENTER to select and ESCAPE to return.

Figure F-3 Reports Menu

The fifth option is **Filing Report (Shelf-List)**. When selected, by pressing <ENTER>, the filing report program is executed. That is, all of the records are shown in the order by which they are filed, the terminal digit filing system.

The sixth option on this menu is **HIV Report**. When selected, by pressing <ENTER>, the HIV report program is executed. All personnel who have not had an HIV test within the last year are listed.

The seventh option on this menu is **NPGS personnel lacking a code**. When selected, by pressing <ENTER>, a report program is executed which displays the names of those individuals who do not have their HIV code in the database.

Reports	Update Data	Exit
Physicals Needed for PFT Checkout Reports Medical Readiness Reports Morbidity Report Filing Report (Shelf-list) HIU Report MPFS personnel lacking a co	Physical exam due today for PFT PE due for PFT (Queary for month) PE due for PFT (Queary for curriculum)	

Use the arrow keys to highlight, ENTER to select and ESCAPE to return.

Figure F-4 Physical Exams for PFT Reports

1. Physicals Needed for PFT

Physicals Needed for the PFT is the first item in the Reports Menu. When selected by pressing <ENTER> an additional menu is displayed. Figure F-4 is an illustration of this menu. The options on this menu can be highlighted with the arrow keys and selected by pressing <ENTER> while the desired option is highlighted. Deselecting this menu is accomplished by pressing <ESC> or <Escape>.

The first option on this menu is **Physical exam due today for PFT**. When selected, by pressing <ENTER>, a PFT report program is executed which displays pertinent information on individuals who do not have current physical exams for the PFT.

The second option on this menu is **PE due for PFT (Query for month)**. When selected, by pressing <ENTER>, a PFT report program is executed which displays pertinent information on individuals who will not have current physical exams for the PFT in the month chosen by the operator.

The third option on this menu is **PE due for PFT (Query for Curriculum)**. When selected, by pressing <ENTER>, a PFT report program is executed which displays pertinent information on individuals who will not have current physical exams for the PFT in the month chosen by the operator.

2. Checkout Reports

Checkout Reports is the second item in the Reports Menu. When selected, by pressing <ENTER>, an additional menu is displayed. Figure F-5 is an illustration of this menu. The options on this menu can be highlighted with the arrow keys and selected by pressing <ENTER> while the desired option is highlighted. Deselecting this menu is accomplished by pressing <ESC> or <Escape>.

The first option on this menu is **Marked Records Report**. When selected, by pressing <ENTER>, a report program is executed which displays the names of individuals whose records are marked for deletion.

The second option on this menu is **Checkouts as of today**. When selected, by pressing <ENTER>, a report program is executed which displays pertinent information on individuals who have check-out dates before the date of the report.

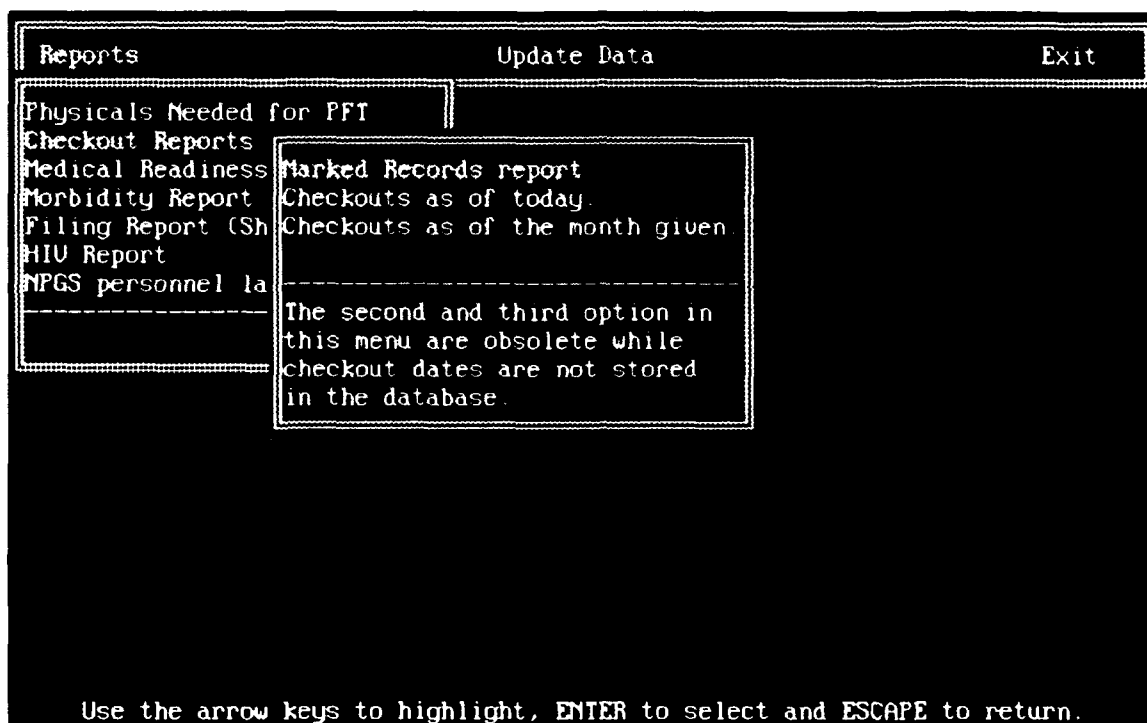


Figure F-5 Check-out Reports

The third option on this menu is **Checkouts as of the month given**. When selected, by pressing <ENTER>, a report program is executed which displays pertinent information on individuals whose check-out dates are prior to the date chosen by the operator.

3. Medical Readiness Reports

Medical Readiness Reports is the Third item in the Reports Menu. When selected by pressing <ENTER> an additional menu is displayed. Figure F-6 is an illustration of this menu. The options on this menu can be highlighted with the arrow keys and selected by pressing <ENTER> while the desired option is highlighted. Deselecting this menu is accomplished by pressing <ESC> or <Escape>.

Reports	Update Data	Exit
Physicals Needed for PFT Checkout Reports Medical Readiness Reports Morbidity Report Filing Report (Shelf-list) HIU Report MPGS personnel lacking a co	Medical Readiness for time period given Medical Readiness as of today Medical Readiness Unit Report One Stop Check-out Readiness Report	

Use the arrow keys to highlight, ENTER to select and ESCAPE to return.

Figure F-6 Readiness reports

The first option on this menu is **Medical Readiness for time period given**. When selected, by pressing <ENTER>, a readiness report program is executed. This report displays all the readiness data for individuals who are delinquent in at least one readiness field for the time window provided by the operator.

The second option on this menu is **Medical Readiness as of today**. When selected, by pressing <ENTER>, a readiness report program is executed. This report displays all the readiness data for individuals who are delinquent in at least one readiness field as of the date of the report.

The third option on this menu is **Medical Readiness Unit Report**. When selected, by pressing <ENTER>, a readiness

report program is executed. This report displays all the readiness data for individuals who are delinquent in at least one readiness field prior to the date provided by the operator. This report option displays only the individuals from the curricular office (curric_div) assigned by the operator.

The fourth option on this menu is **One Stop Check-out Readiness Report**. When selected, by pressing <ENTER>, a readiness report program is executed. This report displays all the readiness data for individuals who are delinquent in at least one readiness field and are marked for deletion. This report option displays only the individuals from the command assigned by the operator.

B. UPDATE DATA

Figure F-7 is an illustration of the Updates menu. The options on this menu are all related to the maintenance of the database. Each option in the menu activates a program which executes one or more functions which permits the operators to perform database maintenance tasks easily.

The last option on this menu, **Delete checkouts** is only useful if accurate checkout data is able to be maintained. **Mark records for deletion, Unmark records for deletion, and Delete marked records** when used in conjunction also assist in check-out database management. "Mark records for deletion" marks records for deletion identified by social security number (SSN). "Unmark records for deletion" un-marks errantly

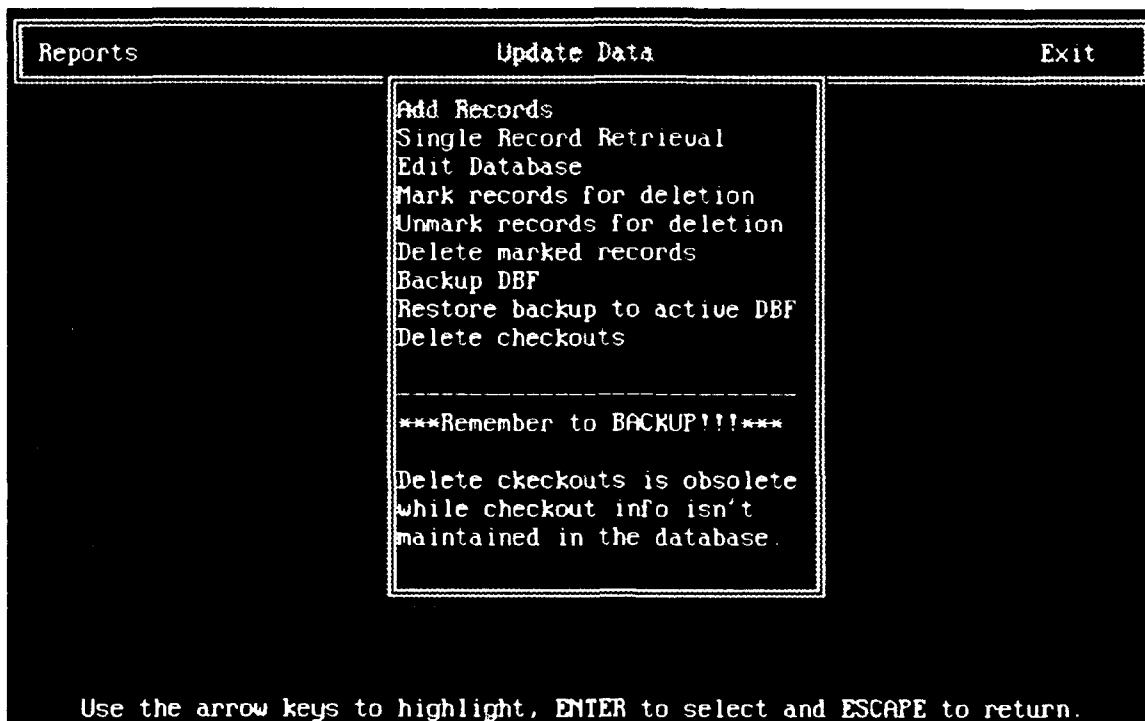


Figure F-7 Update Data

marked records marked for deletion also identified by SSN. "Delete marked records" purges marked records from the database and packs the remaining records.

The sixth and seventh options on this menu expedite some of the back up procedures needed in database maintenance. **Backup DBF** creates a duplicate data file for NEW_DATA.DBF called NEWDATBK.DBF. **Restore backup to active DBF** copies the former NEWDATBK.DBF back-up copy to the active database file NEW_DATA.DBF. This is helpful if the active database file was inadvertently altered.

Figure F-8 is an illustration of the data input program that appears when **Add Records**, the first item on the Update Data menu, is selected. Data input is made easy through

multiple choice options and default values. If a duplicate SSN is entered the error message; "Editing condition not satisfied (Press SPACE)" will appear. The entry for individuals allergic to an immunization is "11/11/11" which is also the default value for these fields. Deselecting the data entry form is accomplished by pressing <ESC> or <Escape>.

The second option on the update data menu is **Single Record Retrieval**. When selected, by pressing <ENTER>, a program is executed which retrieves specific records, identified by SSN, for browse or edit. Returning to the Update Data menu is accomplished by pressing <ESC> or <Escape>.

Records Organize Go To Exit									
SSN									
LASTNAME			FIRSTNAME			INITIAL			
RANK		SERVICE		DOB		COMMAND		CURRIC/DIU	
		BLOOD TYPE		G&PD		SICKLE CELL			
FOX	POL	FFD	WF	TD	TYF	ESC			
LAST FE			HIU			AUDIO			
FLIGHT	DIVE	HALO	RAD	ASBESTOS	RESPIRATOR	NONION RAD			
DATA ENTRY									

Figure F-8 Data Entry Form

The third option on the update data menu is **Edit Database**. When selected, by pressing <ENTER>, the database is retrieved in the browse or edit mode with the first record in the database in view. Returning to the Update Data menu is accomplished by pressing <ESC> or <Escape>.

C. EXIT MENU

Figure F-9 is an illustration of the exit menu. The option on this menu can be highlighted with the arrow keys and selected by pressing <ENTER> while the desired option is highlighted. Selecting "Exit" will return the operator to the operating system. Returning to the main menu is accomplished by pressing <ESC> or <Escape>.

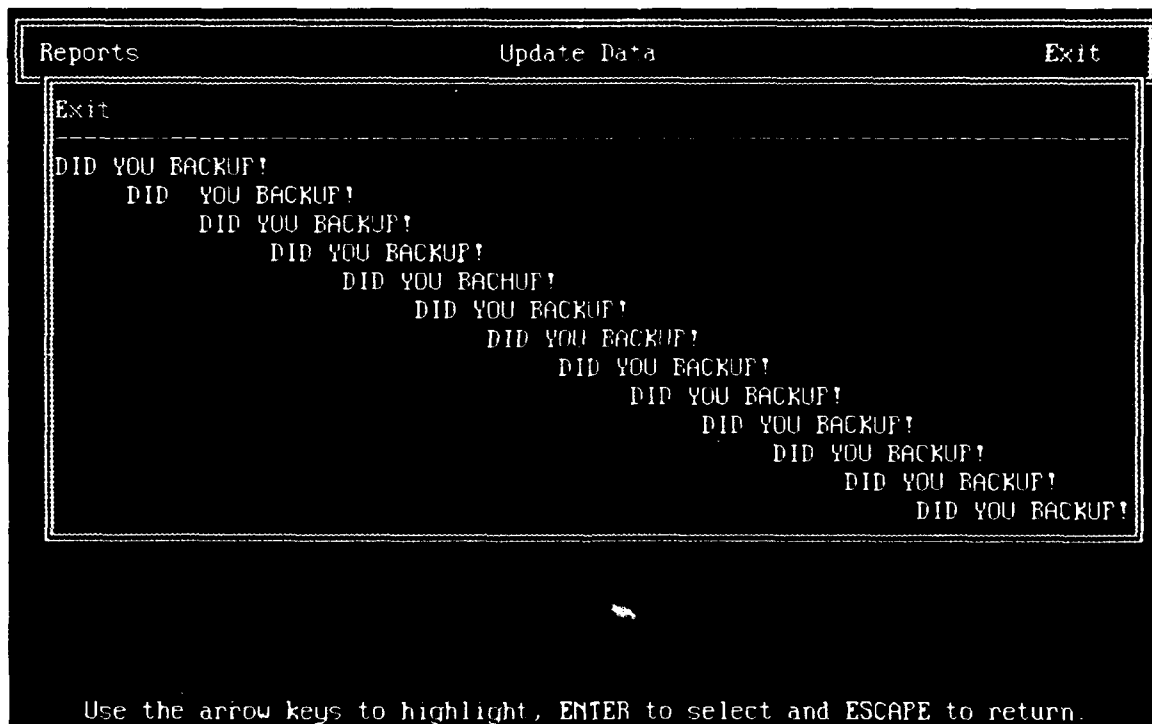


Figure F-9 Exit Menu

IV. BACK-UP DATABASE

The back-up procedures previously mentioned are useful but not inclusive. A back-up copy of the database should be updated daily to an external storage medium. This can be done by typing "copy C:\dbase\NEW_DATA.DBF [drive containing storage disk]:" at the hard drive prompt (usually c:>).

LIST OF REFERENCES

1. Hansen, G. W., Hansen, J.V., *Database Management and Design*, Prentice-Hall, Inc., 1992.
2. Whitten, J. L., Bentley, L. D., Victor, B. M., *Systems Analysis & Design Methods*, Richard D. Irwin, Inc., 1989.
3. Ashton-Tate, *Programming in dBASE IV*, Borland International, Inc., 1992.
4. Mynatt, B. T., *Software Engineering with Student Project Guidance*, Prentice-Hall, Inc., 1990.
5. Pfleeger, C. P., *Security in Programming*, Prentice-Hall, Inc., 1989.
6. Inmon, W. H., *Management Control of Data Processing*, pp. 27-34, Prentice-Hall, Inc., 1983

BIBLIOGRAPHY

Ashton-Tate, *Getting Started*, Boreland International, Inc., 1992.

Ashton-Tate, *Language Reference*, Boreland International, Inc., 1992.

Ashton-Tate, *Using dBASE IV*, Boreland International, Inc., 1992.

Cowart, R., *The ABC's of dBASE IV 1.5*, SYBEX Inc., 1992.

Page-Jones, M., *The Practical Guide to Structured Systems Design*, Prentice-Hall, Inc., 1988.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria VA 22304-6145	2
2. Library, Code 052 Naval Postgraduate School Monterey CA 93943-5002	2
3. Professor Shue Liao, Code AS/LC Naval Postgraduate School Monterey, CA, 93943-5002	2
4. Doctor Richard E. Oswald 16102 Blue Mesa Ridge Drive Friendswood, TX, 77546	1
5. Navy Medical Administrative Unit, Code 04M Building #422 Presidio of Monterey Monterey, CA., 93944-5012	1
6. Kevin Albert Bianchi 11 Winthrop Place Maplewood, N.J., 07040	1